INTELIGÊNCIA ARTIFICIAL E CHATGPT

Da revolução dos modelos de IA generativa à Engenharia de Prompt

EDIÇÃO AMPLIADA E ATUALIZADA

FABRÍCIO CARRARO

Casa do Código | alura

Sumário

- ISBN
- Sobre o autor
- Agradecimentos
- Prefácio
- Sobre o livro
- 1. Introdução à Inteligência Artificial
- 2. Como a Inteligência Artificial aprende?
- 3. Modelos de linguagem e como funcionam
- 4. Como funcionam o ChatGPT, Gemini e outros LLMs
- 5. Agentes de IA
- 6. Parâmetros de calibragem dos LLMs
- 7. Engenharia de Prompt
- 8. Técnicas avançadas com IA no mundo real
- 9. Limitações e considerações éticas sobre a Inteligência Artificial
- 10. Extra — Rodando modelos open source localmente com o LM Studio
- 11. Extra — Programando sistemas em Python com a API do GPT

ISBN

Impresso: 978-85-5519-359-0

Digital: 978-85-5519-358-3

Edição atualizada em julho de 2025.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Carraro, Fabrício
Inteligência artificial e chatGPT : da revolução dos modelos de IA generativa à engenharia de prompt / Fabrício Carraro. -- São Paulo : AOVS Sistemas de Informática, 2023.

Bibliografia.
ISBN 978-85-5519-359-0

1. ChatGPT 2. Ciência da computação 3. Inteligência artificial 4. Tecnologia I. Título.

23-181133 CDD-006.3

Índices para catálogo sistemático:

1. Inteligência artificial 006.3

Aline Graziele Benitez - Bibliotecária - CRB-1/3129

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

Sobre o autor

Fabrício Carraro nasceu em 1988 no ABC paulista. Sua formação compreende Engenharia da Computação pela UNICAMP e a Pós Tech de *Data Analysis* & *Machine Learning* pela FIAP. Atuou por anos na área de desenvolvimento de software, tendo passado por empresas como Griaule Biometrics e UOL PagSeguro.

Sua paixão pelo estudo de línguas estrangeiras o levou a mudar-se para a Alemanha e, posteriormente, para a Espanha, onde vive em Barcelona, mas considera-se um cidadão do mundo, já tendo visitado ou morado em cerca de 40 países. Atualmente, pode se comunicar em 15 idiomas em diferentes níveis, sendo eles o português, inglês, italiano, espanhol, russo, polonês, francês, alemão, grego, turco, croata, romeno, hebraico, holandês e catalão.

Atua como *Program Manager* e especialista de IA na Alura, onde lidera projetos voltados para Inteligência Artificial e educação tecnológica; e também como *AI Developer Advocate* no Barcelona Supercomputing Center.

Fabrício é *host* do podcast *IA Sob Controle* (https://www.iasobcontrole.tech/), com foco em *Machine Learning* e Inteligência Artificial, que é o #1 do Brasil na categoria de "Tecnologia" segundo o Spotify; e do podcast *Dev Sem Fronteiras*, onde entrevista brasileiros e brasileiras que moram e trabalham na área de TI fora do país (https://www.devsemfronteiras.tech/).

Foi líder de criação de produtos como o *TechGuide.sh* (https://techguide.sh/), o *7 Days of Code* (https://7daysofcode.io/), das semanas de Imersão da Alura em parceria com o Google, bem como a Websérie de IAs Generativas (https://www.youtube.com/playlist?list=PLh2Y_pKOa4Ud316ih975nbh3YbF5R4uZP).

É membro da *Google AI Community Brasil*, *Top Voice* em Inteligência Artificial no *LinkedIn* e reconhecido pela *Favikon* como uma das 200 pessoas mais influentes do mundo em IA na plataforma.

Agradecimentos

Primeiramente, aos meus pais, Izabel e Leônidas, pelo esforço, dedicação e amor com o qual me criaram, e por terem sempre me passado a mensagem de que a educação, a ciência e a bondade para com os outros são a chave para sermos pessoas melhores. Amo vocês.

Ao meu avô José, que foi um segundo pai e é, provavelmente, a pessoa mais bondosa que já conheci nesta vida. À minha madrinha Maria, uma segunda mãe e sempre fonte de carinho e diversão. À minha irmã Giuliana, pelo companheirismo e orgulho que me dá vendo seu esforço e dedicação aos estudos. Amo todos vocês.

A Jomar Silva, que escreveu o prefácio deste livro e tem sido uma inspiração há mais anos do que eu posso contar.

Na parte técnica, agradeço a Sthefanie Monica Premebida, Cientista de Dados Sênior; a Filipe Lauar, Engenheiro de *Machine Learning* e criador do podcast "Vida com IA"; e a Guilherme Silveira, *Chief Education Officer*. Este livro é melhor graças a vocês.

A Paulo Silveira, cuja visão e confiança desde 2013 tornaram possível a escrita deste livro em 2023.

Por fim, à Casa do Código e, principalmente, a Vivian Matsui, pela oportunidade de tornar este sonho realidade e pelo doce acompanhamento e apoio durante o processo de escrita.

Prefácio

Quando decidi (na verdade, precisei) aprender Inteligência Artificial, existiam basicamente duas formas, que ainda estão disponíveis — iniciar pela matemática e computação, ou iniciar de forma empírica, seguindo tutoriais e desenvolvendo pequenos projetos que, com o passar do tempo, se tornam mais complexos, e todos os conceitos centrais da tecnologia são absorvidos em doses homeopáticas.

Já fazia mais de uma década que eu tinha me formado em engenharia e, por algum instinto masoquista, decidi começar pela matemática, para desenferrujar os neurônios. Claro que deu bem errado e, algumas semanas depois, eu joguei a toalha: aquilo era ciência de foguete para mim. Eu não conseguia sequer entender os termos usados nos artigos que lia, e muito menos formular buscas que me ajudassem a entender (este é, aliás, um dos motivos de ser tão fã e usuário do *ChatGPT* hoje em dia, porque muitas vezes, aprender a formular perguntas de forma adequada é tudo o que precisamos para aprender algo novo).

Alguns meses depois, a necessidade apertou ainda mais, e me vi realmente obrigado a me dedicar ao tema, mas dessa vez, decidi ir pelo outro caminho, provavelmente mais demorado, mas com maiores chances de dar certo... E deu!

Consegui encontrar alguns cursos bem introdutórios sobre o assunto, mas ainda assim, eu precisei fazer um trabalho inicial de pesquisa para poder entender quais eram as áreas da IA, qual a aplicabilidade de cada uma delas aos problemas que eu precisava resolver, para, finalmente, começar a buscar e fazer cursos, assistir a vídeos, seguir tutoriais e iniciar a minha jornada de aprendizado. Tudo de que eu precisava neste início era que alguém me ajudasse a entender o panorama mais amplo de IA, entender os principais fundamentos, termos, conceitos, tecnologias envolvidas e áreas de aplicação, para, finalmente, com a imagem toda em mente, decidir por onde começar. Levei algumas boas semanas para conseguir ter

esta ideia geral, ainda que muito equivocada, depois de ler muita coisa (e quase desistir novamente). Este guia que eu tanto procurava está neste momento em suas mãos. 🙂

Com linguagem extremamente acessível, exemplos do mundo real e analogias extremamente didáticas, o Fabrício Carraro nos guia por este mundo de tecnologia que é a Inteligência Artificial, desmistificando a "ciência de foguete", que ainda pode ser a concepção do tema para muitos de nós.

Nos últimos anos, com Python se tornando a linguagem central para o desenvolvimento em Inteligência Artificial, é cada vez mais fácil implementar sistemas complexos utilizando algoritmos de computação paralela e distribuída sem que seja necessário ter um Mestrado ou Doutorado nessas áreas. Quando falo sobre a importância de processamento acelerado, paralelizado e distribuído, gosto sempre de lembrar que o modelo *GPT-3.5* (por trás da primeira versão do *ChatGPT*) foi treinado por uma legião de 10.000 *GPUs*.

Constantemente me perguntam em eventos e palestras: "Por onde devo começar a estudar IA?". Sempre recomendo um *overview* da tecnologia e suas aplicações, que é exatamente o que este livro entrega e, depois disso, que se concentre na área que mais lhe atrai ou que mais se aplica ao problema que quer ou precisa resolver. Aí, é mãos à obra.

Uma segunda recomendação é que crie um projeto real para poder aplicar a tecnologia. "Estudar por estudar" não nos permite absorver tanto conhecimento como quando colocamos a mão na massa. Portanto, crie um projeto para o mundo real, aplique o que está aprendendo para construí-lo, e você verá que, no final do processo, além de ter aprendido bastante, já vai ter o seu primeiro projeto desenvolvido e aplicado a um problema real do seu dia a dia.

Se não pensa em nenhum problema específico, existem diversos sites na Internet, como o *Kaggle* (https://www.kaggle.com/), que

propõe desafios constantemente e, neste caso, além de aprender, você ainda pode se juntar a um grupo de pessoas que estão aprendendo como você e, certamente, com pessoas que estão ali exatamente para compartilhar o conhecimento que têm. O conhecimento contraria a matemática, porque o multiplicamos a cada vez que dividimos.

Espero que esta leitura seja tão prazerosa para você como foi para mim, e que, ao final dela, esteja motivado a continuar seus estudos e tenha encontrado a área que mais lhe atrai.

Que a sua jornada de aprendizado seja tão divertida e com resultados concretos como está sendo a minha — e não se esqueça: compartilhe sempre o que aprender durante ela!

Jomar Silva

Sobre o livro

A origem da Inteligência Artificial vem das décadas de 1950 e 60, mas os anos de 2022 e 2023 viram uma revolução, com a chegada e popularização em tempo recorde das IAs generativas como o *ChatGPT*, o *Gemini*, o *Midjourney* e muitas outras.

Se você acha esse tema fascinante, assim como eu, mas não sabe exatamente por onde começar a estudá-lo e quer entender como essas ferramentas fantásticas funcionam por baixo dos panos, este livro foi feito para você.

No capítulo 1, explicarei a base sobre o que é a Inteligência Artificial, seus diferentes tipos e classificações, e como ela já está muito presente no seu dia a dia mesmo sem você perceber.

No capítulo 2, entrarei na parte técnica, mas ainda de maneira suave e didática, para lhe explicar como as Inteligências Artificiais aprendem. Ao final, você entenderá os conceitos de *Machine Learning*, os diferentes tipos de aprendizado (Supervisionado, Não Supervisionado, Autossupervisionado, Por Reforço etc.), redes neurais artificiais e *Deep Learning* e terá a base necessária para compreender qualquer sistema de IA moderno.

No capítulo 3, demonstrarei o que é um modelo de linguagem, como eles funcionam, e contarei sobre a evolução da área e dos algoritmos de PLN (Processamento de Linguagem Natural), como N-grama, *word embeddings*, redes neurais profundas, os famosos *Transformers* que possibilitaram a criação de *LLMs* como o *ChatGPT* e o *Gemini*, possíveis novas arquiteturas para o futuro e o *Small Language Models*.

O capítulo 4 será totalmente dedicado aos *LLMs*, com um foco inicial no treinamento do primeiro *ChatGPT*. Veremos com profundidade o que eles fazem, como funcionam por dentro, como os seus modelos foram treinados, além de abordar em tópicos como *tokens*, temperatura, alucinações, as diferenças dos modelos originais para o

GPT-4 e *GPT-4o*, a arquitetura de *Mixture-of-Experts*, os modelos de raciocínio e híbridos, o uso de ferramentas pelos *LLMs* (*function calling*) e a *Deep Research*.

O capítulo 5 é totalmente novo na edição atualizada e ampliada deste livro e mergulha de cabeça no que muitos consideram o próximo grande salto: os agentes de IA. Explicarei o que são, como funcionam, a diferença entre agentes de IA e sistemas multiagentes, os protocolos MCP e A2A, além de introduzir os principais *frameworks* do mercado para a construção de sistemas de agentes robustos.

No capítulo 6, esmiuçarei um a um os parâmetros de calibragem oferecidos no *Playground* da *OpenAI* e via API para qualquer modelo, para o uso programático de *LLMs*.

No capítulo 7, mergulharemos no mundo da Engenharia de *Prompt*, explicando os princípios mais importantes e as técnicas mais relevantes (*Few-shot Prompting*, *Chain-of-Thought* etc.) desenvolvidas por pesquisadores de empresas de IA e universidades do mundo todo a fim de gerar *prompts* ideais, transformando você de um usuário casual em um operador avançado, capaz de extrair o máximo de poder e precisão dos *LLMs*.

No capítulo 8, explicarei o que são e como funcionam algumas das técnicas avançadas de IA mais usadas na prática no mundo real por desenvolvedores e pesquisadores de IA, como *RAG*, suas técnicas (*reranking*, *Context Retrieval*) e variações (*GraphRAG*, *CAG*); quantização; *fine-tuning* com *LoRA* e *QLoRA*, e destilação.

No capítulo 9, talvez o mais importante do livro, falarei de temas relacionados às limitações e à ética sobre a Inteligência Artificial. Aqui você se debruçará sobre tópicos que estão sendo discutidos exaustivamente em fóruns mundiais sobre IA, como as questões de privacidade, segurança, regulamentação, geopolítica, direitos autorais, *fake news*, vieses, alinhamento, uso de água e energia, e as possíveis repercussões no mercado de trabalho.

O capítulo 10 será um "extra" extremamente útil, que ensinará você a rodar modelos *open source* de forma local na sua máquina e a criar uma *API* local com esses próprios modelos, que você pode baixar livremente, tudo com a ferramenta *LM Studio*.

Para finalizar, o capítulo 11 será mais um "extra", este apenas para pessoas desenvolvedoras ou para quem esteja aprendendo a programar, onde ensinarei como desenvolver um sistema de *chatbot* inteligente utilizando a linguagem de programação Python conectada à API da *OpenAI* e aos modelos *GPT*.

Certamente, este livro pode ser usado como ferramenta em disciplinas introdutórias que trabalham os conceitos de Inteligência Artificial, *Machine Learning*, redes neurais, *Deep Learning* e Processamento de Linguagem Natural com modelos de linguagem. Todas as referências bibliográficas estão descritas ao final de cada capítulo, caso você deseje se aprofundar ainda mais nos temas.

Que a leitura deste livro seja tão intelectualmente empolgante para você quanto o processo de escrita dele foi para mim. Aproveite para mergulhar nesse assunto, e espero que, ao final, você também se apaixone pelo mundo da Inteligência Artificial.

Vamos começar?

Capítulo 1
Introdução à Inteligência Artificial

1.1 O que é Inteligência Artificial?

Meu primeiro pedido para você é: não se assuste! Eu sei que esse nome **Inteligência Artificial** parece à primeira vista algo mágico, virtualmente inexplicável. Acredito que essa sensação tenha muito a ver com a palavra "inteligência", algo que nós humanos ainda não conseguimos realmente definir.

Conseguimos atribuir facilmente o conceito de inteligência a um chimpanzé que memoriza números e linguagem de sinais, ou a um polvo que consegue abrir uma caixa para pegar a comida que está dentro, mas o que dizer sobre fungos que conseguem transformar formigas em *zumbis* controlados, levá-las até uma altura específica de uma planta e fazer chover seus esporos a fim de se espalharem pela área? E sobre o bolor limoso, um ser similar a uma ameba, que não possui cérebro, mas consegue resolver facilmente problemas complexos de grafos, como encontrar o caminho mais curto entre dois pontos de um labirinto?

E sistemas computacionais que realizam operações complexas? Também podemos atribuir o conceito de inteligência a eles?

Uma das pessoas que começaram a discussão sobre a inteligência das máquinas foi o matemático e cientista da computação Alan Turing. Em seu artigo científico "*Computing Machinery and Intelligence*"[1] , ele discute a questão da construção de máquinas inteligentes e sobre como testar sua inteligência. Nesse artigo, ele propõe o famoso *Jogo da Imitação*, também conhecido como *Teste de Turing*, no qual temos, de forma resumida, um ser humano e uma máquina tentando convencer um juiz humano sobre quem desses dois seria verdadeiramente o ser humano. Porém, Turing não

tinha capacidade computacional ou financeira para colocar suas propostas à prova.

Essa história começou a mudar em 1956. John McCarthy, um professor assistente do Dartmouth College, decidiu criar um grupo de pesquisadores sobre o tema crescente de "máquinas pensantes". Ele organizou uma conferência chamada "Projeto de Pesquisa de Verão sobre Inteligência Artificial de Dartmouth", que foi quando o termo foi finalmente cunhado. Esse evento, apesar de pequeno, criou a fagulha necessária para a explosão de pesquisas nesse campo que se seguiu nas próximas décadas, que passou por capítulos como a vitória do computador *Deep Blue* da IBM sobre o Grande Mestre de xadrez Gary Kasparov e veio a culminar mais recentemente na popularização dos *LLMs* (*Large Language Models*), como o ChatGPT, que bateu o recorde de "tecnologia mais rapidamente adotada da história", obtendo mais de 100 milhões de usuários em apenas dois meses (o recorde anterior era do aplicativo TikTok, que havia levado nove meses).

Porém, apesar de ser um tema interessante e que será abordado esporadicamente nos próximos capítulos, este não é um livro sobre a história da Inteligência Artificial.

Queríamos responder à pergunta que dá nome a esta seção, "O que é Inteligência Artificial?", certo? Uma resposta oficial provavelmente soaria como: *A Inteligência Artificial (IA) é um campo da Ciência da Computação que se concentra na criação de sistemas e programas de computador que podem realizar tarefas que normalmente exigiriam inteligência humana. Esses sistemas são capazes de aprender, raciocinar, tomar decisões e resolver problemas de maneira semelhante aos seres humanos.*

E sim, essa definição acima foi criada por uma ferramenta de IA, o famoso **ChatGPT**, usando o modelo *GPT-4* (abordaremos adiante como esses modelos funcionam).

Porém, se preferir uma explicação mais "mão na massa", podemos dizer que são sistemas criados usando diferentes tecnologias para realizar tarefas que (em teoria) apenas uma inteligência humana poderia realizar, como a capacidade de raciocínio lógico, memória e o aprendizado com generalização (aprender com situações passadas para tomar melhores decisões no futuro) e inferência ("*A Maria está em casa ou na escola. Eu cheguei em casa e ela não está, portanto ela deve estar na escola*").

A maneira como boa parte deles funciona é: depois de serem alimentados com um grande número de dados específicos de determinada área, analisam e categorizam esses dados para tentar encontrar padrões — às vezes com um juiz (humano ou algoritmo de máquina) dando um *feedback* sobre se essa categorização foi satisfatória ou não, outras vezes apenas tentando inferir por conta própria com base nos dados alimentados — e, finalmente, usam esses padrões para tentar prever ou saber como agir em estados futuros, com dados para os quais eles não foram treinados.

Tenho um amigo que diz que um nome mais honesto para Inteligência Artificial seria "*Métodos Matemáticos e Estatísticos Avançados*". Concordando ou não com ele, há de se admitir: mais honesto ou não, esse nome seria extremamente mais entediante do que o atual.

Mas claro, essas tecnologias são a base para a criação de modernas ferramentas de IA, que podem ser usadas em uma infinidade de aplicações, como reconhecimento de fala, processamento de linguagem natural, visão computacional, controle de veículos autônomos, geração de imagens a partir de texto... As possibilidades são incontáveis.

E esse é um dos intuitos dos próximos capítulos: tentar torná-las um pouco mais contáveis, explicando para você como essas tecnologias funcionam por baixo dos panos, além de demonstrar como elas estão sendo utilizadas em diferentes áreas do conhecimento e

discutir as melhores técnicas para usá-las na prática no seu dia a dia de trabalho.

1.2 Inteligência Artificial no seu dia a dia

Muitas pessoas com menos contato com a área podem pensar que sistemas com base em Inteligência Artificial começaram a ser realmente usados de forma abrangente entre o final de 2022 e início de 2023, coincidindo com o lançamento do *ChatGPT* ao público. Esse lançamento foi certamente um ponto de disrupção, do contrário, não teria tomado conta das capas de jornais e páginas com tanto furor. Porém, esses sistemas têm feito parte de nossas vidas há muito tempo, e é isso que eu quero lhe mostrar nos próximos parágrafos.

Um exemplo bastante conhecido é o uso da Inteligência Artificial em sistemas de classificação e recomendação de mídia, aplicada para filmes, vídeos ou música. Assinantes da *Netflix*, o maior serviço de streaming de filmes e séries do mundo, são afetados diariamente: cada filme ou série a que assistem até o final ou que param na metade, cada nota "gostei" ou "não gostei" que dão etc., são considerados pelo algoritmo de IA da plataforma para recomendar a eles de maneira personalizada outros filmes e séries a que eles tenham uma maior probabilidade de querer assistir e, consequentemente, passar mais tempo na plataforma, mantendo a crença de que vale a pena continuar sendo assinante do serviço por mais um mês. Até mesmo as *thumbnails* (imagens de capa) são sugeridas de maneira personalizada, considerando qual das opções disponíveis tem uma chance maior de chamar a atenção daquela pessoa em específico.

Esses sistemas não são uma exclusividade na *Netflix*. Todas as grandes empresas do ramo utilizam há anos algoritmos similares baseados em IA, como a *Amazon Prime Video*, a *Disney+*, a *Apple TV* etc. Cerca de 75% dos filmes e séries assistidos nessas

plataformas são devido a seus sistemas de recomendação, segundo relatório da *McKinsey & Company*.[2]

A *Spotify* é um outro bom exemplo, que utiliza informações sobre as músicas que você mais ouve e que sinaliza como "favoritas" para gerar novas *playlists* (listas de reprodução), como a de "Descobertas da Semana", entre outras, que são pré-selecionadas com base no que o sistema sabe que você gosta e também em músicas que outras pessoas com gosto musical similar ao seu apreciam.

A gigante do setor de comércio virtual *Amazon* também é muito reconhecida pelo uso de IA em seu algoritmo. Com base não só nos seus dados demográficos, como idade, gênero, localização etc., mas também em compras passadas feitas por você na plataforma, o modelo de predição consegue lhe recomendar produtos que podem potencialmente ser de seu interesse, o que gera mais vendas. O relatório da *McKinsey & Company* já mencionado calculou que, em média, 35% de todas as compras feitas na Amazon foram geradas por seu sistema de recomendações. A chinesa *Alibaba* tem afirmações similares.[3]

A Inteligência Artificial está por trás dos sistemas de voz da *Alexa* (Amazon), *Siri* (Apple), *Google Assistant* (Google) e *Bixby* (Samsung), que entendem as perguntas e comandos de voz dados por seres humanos, seu contexto, e (no geral) oferecem uma resposta apropriada. Esses sistemas utilizam conceitos que veremos mais adiante, como *Deep Learning* (Aprendizado Profundo) e PLN (Processamento de Linguagem Natural).

E a lista é enorme. A automotiva *Tesla* utiliza Inteligência Artificial para "ensinar" seus carros autônomos a dirigirem de maneira segura nas vias e a escaparem de possíveis acidentes automobilísticos. A *Meta* o faz para conhecer melhor os usuários de seus serviços *Facebook* e *Instagram* e direcionar-lhes anúncios mais relevantes. Companhias aéreas como a *EasyJet* o fazem para ajudar a definir os preços das passagens aéreas, oferecer promoções personalizadas por email, entre outras coisas.

E uma área onde o uso de tais tecnologias só vem crescendo é a medicina, onde IAs estão cada vez mais sendo usadas para ajudar médicos e médicas a realizarem diagnósticos e, principalmente, para analisar imagens de exames, como de raio-x, de ressonância magnética, entre outros — algo que máquinas treinadas podem fazer com muito mais precisão e velocidade do que humanos. Claro, essas análises são, posteriormente, verificadas por profissionais da saúde para garantir sua corretude.

Ou seja, essa tecnologia tem estado em voga há muitos anos e você já tem sido diretamente afetado(a) por muitas delas. A principal diferença é que, até este momento, tudo isso era feito por baixo dos panos e, geralmente, para um propósito específico, mas agora, *LLMs* (*Large Language Models*) como o *ChatGPT*, o *Gemini* e o *Claude* estão acessíveis a qualquer pessoa e podem ser usados para diferentes objetivos: criar um email educado do zero, traduzir um texto, gerar a base para estratégias de marketing e até mesmo encontrar erros de programação no seu código.

1.3 Tipos de Inteligência Artificial

Para começarmos a nos aprofundar em Inteligência Artificial, vamos começar do básico: em que tipos elas são classificadas. Existem duas maneiras diferentes de classificá-las, uma que toma como base a capacidade das IAs e outra que foca na sua funcionalidade.

Classificação por Capacidade

1. Inteligência Artificial Fraca ou Estreita (ANI)

A sigla ANI vem de *Artificial Narrow Intelligence* (em português, Inteligência Artificial Estreita) e se refere a um tipo de IA que é

projetada e treinada para executar uma tarefa específica. Ela opera sob um conjunto limitado de restrições e não tem a capacidade de compreender ou aprender qualquer coisa além do que foi programada para fazer. Alguns exemplos de Inteligência Artificial Estreita são:

- **Assistentes virtuais**: todos os sistemas mencionados anteriormente, como a *Alexa* (Amazon), *Siri* (Apple), *Google Assistant* (Google) e *Bixby* (Samsung) são exemplos de IA Fraca ou Estreita. Eles podem responder às perguntas do usuário, realizar tarefas como definir lembretes ou reproduzir música, e até mesmo interagir em um diálogo básico, mas são estritamente limitados às funções para as quais foram programados.

- **Sistemas de recomendação**: como mencionado, sistemas de recomendação como os da *Netflix*, *Spotify*, e *Amazon* usam algoritmos de IA para analisar o comportamento dos usuários e fazer recomendações personalizadas. No entanto, esses sistemas não são capazes de fazer julgamentos independentes ou compreender o conteúdo que estão recomendando além dos parâmetros estabelecidos pelos seus dados de treinamento.

- **Veículos autônomos**: carros autônomos como os da *Tesla* podem navegar e tomar decisões com base em seus sensores e dados de entrada. Eles podem reagir ao ambiente, como parar em um sinal vermelho ou desviar de um obstáculo na estrada, mas não têm consciência do que estão fazendo ou por quê.

- **Softwares de reconhecimento de voz**: aplicações como o *Google Tradutor* podem transcrever a fala humana com precisão, mas não têm compreensão do significado do que estão transcrevendo.

- **Filtros faciais**: esses filtros utilizam técnicas de Inteligência Artificial, como o Aprendizado de Máquina, para analisar o seu rosto e criar uma versão alternativa, seja mais velha, mais

jovem ou do gênero oposto. Eles são treinados com uma grande quantidade de imagens e conseguem aplicar essas transformações ao rosto que está sendo analisado no momento.

Por mais úteis que esses sistemas possam ser, eles são focados em apenas uma (ou poucas) tarefa específica e são altamente especializados e limitados em sua capacidade. Assim, IA Fraca ou Estreita é, como pudemos ver com os exemplos, o tipo mais prevalente de Inteligência Artificial na nossa sociedade atualmente. Porém, não se engane pelo nome "*IA Fraca*" — isso não é sinônimo de ruim ou com baixo desempenho, apenas se trata de uma IA que lida muito bem com um tipo de problema específico, mas não consegue atuar em outras áreas sem ser remodelada.

2. Inteligência Artificial Forte ou Geral (AGI)

A sigla AGI vem de *Artificial General Intelligence* (em português, Inteligência Artificial Geral) e se refere a uma IA hipotética que teria um tipo de inteligência similar à humana, com a capacidade de entender, aprender, adaptar-se e implementar conhecimento como nós fazemos. Ao contrário da IA Fraca, ela não seria limitada a uma tarefa específica, mas teria a habilidade de aplicar a sua inteligência de forma abrangente para resolver problemas desconhecidos, de maneira similar aos humanos.

Uma AGI seria capaz de compreender como objetos se movimentam no espaço físico, além de conceitos mais abstratos, como perceber a passagem do tempo e como ações presentes podem influenciar resultados futuros.

Atualmente, a Inteligência Artificial Forte ou Geral é considerada apenas teórica, não tendo sido alcançada com as tecnologias de IA atuais. É praticamente consenso que mesmo o modelo *GPT-4o* da OpenAI não chega perto do que seria uma AGI, pois não possui alguns dos pontos que mencionei antes: memória persistente de longo prazo, capacidade de aprender de forma contínua e autônoma com novas interações e criação de planos futuros. Porém, vale citar

que, em um artigo científico publicado em abril de 2023, a equipe da *Microsoft Research* afirmou que, dadas as suas capacidades, o modelo *GPT-4* poderia ser visto como uma "*versão inicial (mas ainda incompleta) de um sistema de Inteligência Artificial Geral (AGI)*"[4] , algo que pode ser visto como extremamente entusiasmante para uns, mas preocupante para outros.

Na ficção, temos alguns exemplos de Inteligência Artificial Forte ou Geral, como o androide *Data* da série *Star Trek: The Next Generation* (1987) ou os robôs humanoides da série *Westworld* (2016).

3. Superinteligência Artificial (ASI)

A sigla ASI vem de *Artificial Super Intelligence* (em português, Superinteligência Artificial) e se refere a uma forma hipotética de IA que não apenas igualaria a inteligência humana, mas a ultrapassaria em todos os sentidos e se tornaria consciente de si mesma.

Esse tipo de inteligência seria capaz de superar humanos em praticamente todos os campos, incluindo produção científica, matemática, arte, emoções e relacionamentos interpessoais e amorosos. Devido a sua capacidade de raciocínio e memória altamente superiores às humanas, uma Superinteligência Artificial teria a capacidade de tomar decisões, solucionar problemas complexos, aprender, adaptar-se e entender emoções de uma maneira que nenhum humano poderia.

Assim como a IA Geral, a Superinteligência Artificial ainda vive apenas no reino da teoria e da ficção científica, sendo um passo ainda mais distante do que a AGI. Um exemplo é a IA *Samantha* do filme *Her* (2013), que evolui ao ponto de ter sentimentos como o amor e, eventualmente, transcende a necessidade de interagir com seres humanos.

O ano de 2025 marcou a introdução desse termo na indústria de IA. Ilya Sutskever, uma das mentes mais respeitadas da área e ex-cientista-chefe da *OpenAI*, lançou sua nova empresa, a *Safe*

Superintelligence, que nasceu com o foco extremamente ambicioso de construir uma superinteligência artificial. É importante ressaltar que esse movimento não tornou a *ASI* uma realidade iminente; a questão de se uma superinteligência é de fato alcançável continua a ser um mistério. De qualquer forma, a criação de uma empresa com esse foco por uma das mentes mais brilhantes do mercado transformou essa busca, mesmo sem qualquer garantia de sucesso, de uma abstração utópica em um objetivo de engenharia muito bem financiado.

Classificação por Funcionalidade

As 4 categorias a seguir foram propostas por Arend Hintze[5] , pesquisador e professor de Genética, Biologia Evolutiva e Machine Learning na Dalarna University. Elas são:

1. Máquinas Reativas

As *Máquinas Reativas* são a forma mais básica de Inteligência Artificial, quando falamos em termos de funcionalidade. Elas não têm memória e não são capazes de considerar experiências passadas para tomar ações futuras. Dessa forma, uma determinada entrada terá sempre a mesma saída e, apesar de poderem realizar tarefas complexas, suas ações estão restritas aos parâmetros de sua programação inicial.

Um exemplo de *Máquina Reativa* é o computador *Deep Blue* da IBM, mencionado anteriormente por ter derrotado o Grande Mestre de xadrez Gary Kasparov. O *Deep Blue* foi projetado para identificar as peças no tabuleiro de xadrez e calcular possíveis movimentos futuros, mas não tinha a capacidade de aprender com erros em jogos anteriores ou de adaptar a sua estratégia em tempo real.

2. Inteligência Artificial de Memória Limitada

IAs de Memória Limitada são um tipo de Inteligência Artificial que pode aprender com experiências passadas ou dados novos. Esse tipo

de IA utiliza técnicas de *Machine Learning* (Aprendizado de Máquina) para aprender a partir de tentativas passadas e melhorar sua acurácia a partir de novos conjuntos de dados. Porém, esses novos dados podem ser usados apenas de maneira temporária e não são armazenados indefinidamente na memória da IA.

Exemplos de IA de Memória Limitada incluem:

- **Veículos autônomos**: carros autônomos são um bom exemplo de *IAs de Memória Limitada*. Um carro autônomo precisa lembrar a velocidade e a distância de outros carros na pista para mudar de faixa ou fazer uma curva. Isso é feito através de sensores e câmeras que capturam as situações atuais no trânsito e utilizam essas informações temporárias na tomada de decisões.
- **Chatbots e assistentes virtuais**: a maioria dos chatbots e assistentes virtuais é projetada para ter uma curta memória sobre interações passadas. Por exemplo, se você perguntar à *Alexa* ou ao *Google Assistant*: "*Qual é a previsão do tempo para hoje em São Paulo?*" e, em seguida, após a resposta, você perguntar "*E amanhã?*", o assistente conseguirá "se lembrar" desse contexto de que a palavra "*amanhã*" ainda está se referindo à previsão do tempo em São Paulo. Porém, esses dados não ficarão armazenados para sempre no sistema.
- **Sistemas de recomendação**: os sistemas de recomendação da *Netflix*, *Amazon* e *Spotify* que mencionei anteriormente são exemplos clássicos de IAs de Memória Limitada.

3. Inteligência Artificial com Teoria da Mente

Na Psicologia, o termo *Teoria da Mente* se refere à capacidade de entender outras pessoas atribuindo estados mentais a elas, ou seja, o conhecimento de que as crenças, desejos, emoções e pensamentos do outro podem ser diferentes do seu próprio. Em seres humanos, a nossa própria *Teoria da Mente* se desenvolve entre

os 3 e 4 anos de idade, e geralmente só é considerada completa aos 5 anos de idade[6] .

Em IA, a *Teoria da Mente* se refere a um tipo hipotético de Inteligência Artificial que teria essa capacidade de compreender e interagir com outros indivíduos levando em conta os seus pensamentos e emoções, e usando essa compreensão para definir suas próprias ações e seu comportamento.

Construindo IAs com Teoria da Mente, nós poderíamos, por exemplo, criar cursos que se adaptassem ao estilo de aprendizado e ao estado mental do aluno, levando em conta fatores como estresse, tédio ou confusão. Poderíamos desenvolver também IAs que ajudassem no tratamento de problemas de saúde mental, que saberiam "ler" as emoções nas palavras dos usuários para responder de forma apropriada.

4. Inteligência Artificial Autoconsciente

Por fim, mais um estágio hipotético. Uma *Inteligência Artificial Autoconsciente* seria a concepção mais avançada e complexa de Inteligência Artificial. Além de ter uma compreensão avançada do mundo ao seu redor, ela também teria consciência de si mesma, da mesma maneira como nós, seres humanos, temos consciência de nós mesmos.

Ela não apenas compreenderia as emoções e motivações dos outros, como uma IA com Teoria da Mente, mas também teria suas próprias emoções, motivações e até senso de moralidade. Em outras palavras, uma IA Autoconsciente seria capaz de experienciar o mundo ao seu redor de uma maneira praticamente indistinguível à de um ser humano consciente.

Alguns exemplos de *IA Autoconsciente* na ficção são o androide *Data* da série *Star Trek: The Next Generation* (1987), o robô *HAL 9000* do filme *2001: Uma Odisseia no Espaço* (1968), a IA *Samantha* do filme *Her* (2013) e o *Agente Smith* da série *Matrix*.

Figura 1.1: Tipos de IA - Classificação por Funcionalidade

Ao explorar esse amplo espectro dos tipos de Inteligência Artificial, fica evidente que estamos apenas começando a tatear as possibilidades do que a IA é capaz. Existem desafios significativos até chegarmos à *Superinteligência Artificial* ou à *IA Autoconsciente*, e apesar de existirem pesquisas sérias sendo realizadas nessa área do conhecimento, não existe nem mesmo consenso sobre essa possibilidade ser algo alcançável ou não.

De qualquer forma, o recente *boom* das IAs generativas com o público e a mídia certamente influenciará o desenvolvimento dessas tecnologias, tanto de um ponto de vista de mais dinheiro sendo injetado para pesquisas nesse campo, quanto de mais jovens profissionais e estudantes com interesse em se tornarem pesquisadores na área de IA. O futuro é realmente entusiasmante.

Referências bibliográficas:

[1] TURING, Alan M. I. — Computing Machinery and Intelligence. *MIND*, v. LIX, n. 236, 1950. Disponível em: https://academic.oup.com/mind/article-pdf/LIX/236/433/9866119/433.pdf

[2] MACKENZIE, Ian; MEYER, Chris; NOBLE, Steve. How retailers can keep up with consumers. *McKinsey & Company*, 2013. Disponível em: https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

[3] FAGGELLA, Daniel. Artificial Intelligence at Alibaba – Two Current Use-Cases. *Emerj*, 2022. Disponível em: https://emerj.com/ai-sector-overviews/artificial-intelligence-at-alibaba/

[4] BUBECK, Sébastien; CHANDRASEKARAN, Varun; ELDAN, Ronen; GEHRKE, Johannes; HORVITZ, Eric; KAMAR, Ece; LEE, Peter; LEE, Yin Tat; LI, Yuanzhi; LUNDBERG, Scott; NORI, Harsha; PALANGI, Hamid; RIBEIRO, Marco T.; ZHANG, Yi. Sparks of Artificial General Intelligence: Early experiments with GPT-4. *arXiv*, 2023. Disponível em: https://arxiv.org/abs/2303.12712

[5] HINTZE, Arend. Understanding the Four Types of

AI, from Reactive Robots to Self-Aware Beings. *The Conversation*, 2016. Disponível em: https://theconversation.com/understanding-the-four-types-of-ai-from-reactive-robots-to-self-aware-beings-67616

[6] ROTH, Gerhard; DICKE, Ursula. Evolution of the brain and intelligence in primates. *Progress in Brain Research*, v. 195, p. 413-430, 2012. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/B9780444538604000209

Capítulo 2
Como a Inteligência Artificial aprende?

Neste capítulo, você terá uma visão geral sobre as diversas técnicas através das quais uma Inteligência Artificial aprende. Explicarei o seu funcionamento básico, principais conceitos e abordagens — como o Aprendizado de Máquina (*Machine Learning*) e o Aprendizado Profundo (*Deep Learning*) com redes neurais — e também como esses algoritmos são treinados para tomar decisões.

2.1 Subáreas da Inteligência Artificial

Antes de mergulharmos em como inteligências artificiais aprendem, vale apresentar superficialmente alguns conceitos-chave que serão mencionados neste e nos próximos capítulos como subáreas da Inteligência Artificial. Várias delas, inclusive, não são conceitos excludentes, relacionando-se com as outras. Vamos lá?

- **Machine Learning**: traduzida como "*Aprendizado de Máquina*", está focada em sistemas computacionais que aprendem por conta própria a partir da análise de conjuntos de dados. Teremos uma seção completa dedicada a *Machine Learning*.
- **Deep Learning**: uma subárea do *Machine Learning* que utiliza redes neurais artificiais. Também teremos uma seção completa dedicada ao *Deep Learning*.
- **Processamento de Linguagem Natural**: também conhecido pela sigla *PLN* em português ou *NLP* (*Natural Language Processing*) em inglês, este ramo da IA foca-se em fazer com que as máquinas possam compreender, interpretar e gerar linguagem humana, como o português, o inglês ou mesmo

código de programação. Algumas das aplicações são em tradução automática, análise de sentimento, criação de chatbots "inteligentes" e de resumo de textos. O *ChatGPT* está inserido neste contexto de PLN.

- **Reconhecimento de Fala**: como o próprio nome diz, esses sistemas leem ondas sonoras, reconhecem a fala humana e podem convertê-la em texto ou em comandos para sistemas ativados por voz. Alguns exemplos são os assistentes virtuais, como a *Alexa* e a *Siri*.

- **Visão Computacional**: envolve dar às máquinas a capacidade de interpretar visualmente o mundo através de imagens e vídeos. Suas aplicações abrangem diversos contextos, desde sistemas de reconhecimento facial, biometria e até veículos autônomos, que devem reconhecer o caminho e obstáculos à sua frente.

- **Robótica**: uma IA pode ter contato físico com o mundo real através da robótica, que é uma área próxima da Engenharia Mecatrônica. Também engloba sistemas embarcados, uma área fascinante e acessível a todos através de placas como o *Arduino* e *Raspberry Pi*. É comum na indústria, em armazéns e até na área militar.

Existem outras subáreas que poderiam ser citadas aqui também, mas essas são algumas das principais que você verá nos próximos capítulos.

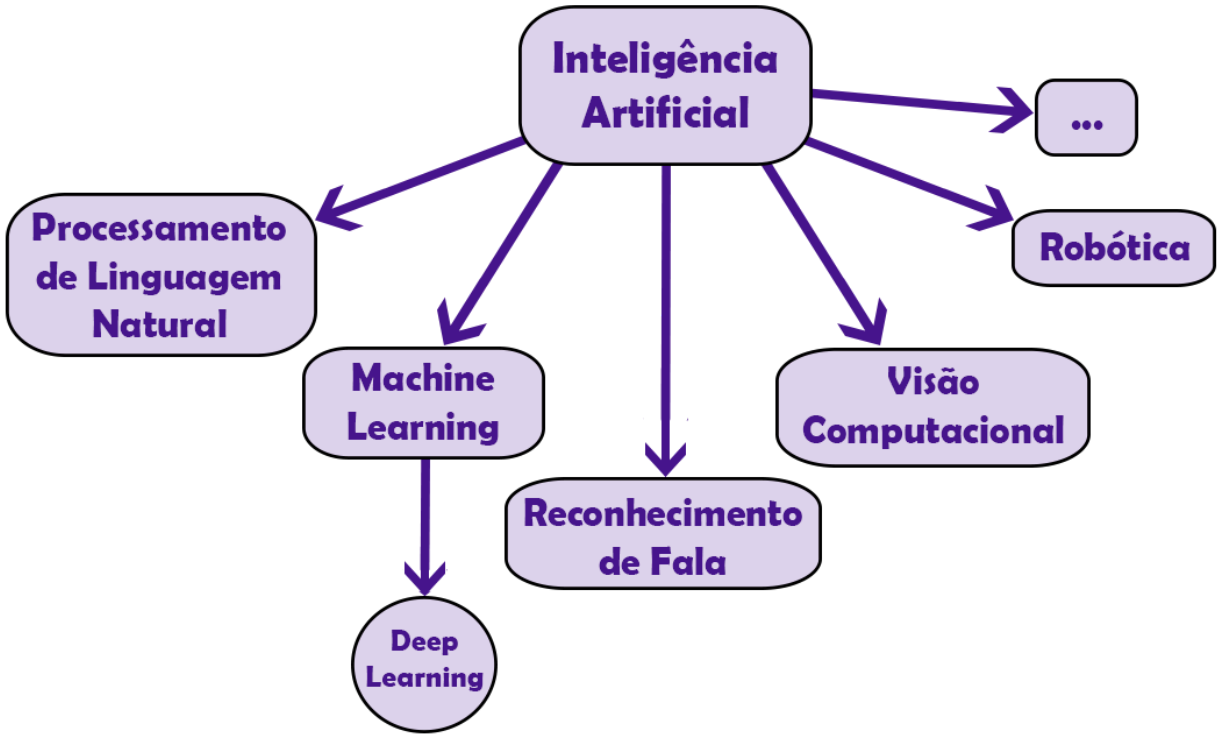

Figura 2.1: Subáreas da Inteligência Artificial

2.2 Visão geral do processo de aprendizado

O processo de aprendizado de uma Inteligência Artificial é fascinante em sua complexidade e, embora possa parecer algo totalmente abstrato e distante, tem mais em comum com nossas formas de aprendizado do que você pode pensar. Por esse motivo, antes de começarmos a discutir os modos como uma IA aprende na prática, vale a pena falarmos um pouco sobre como outros tipos de inteligência funcionam e são "treinados".

Diferentes tipos de animais adquirem habilidades de diferentes maneiras, mas vamos usar o cachorro para o nosso exemplo. Vou chamar nosso cão hipotético de Scooby. O Scooby pode ser ensinado a realizar truques impressionantes, como dar cambalhotas, fingir-se de morto, entre outras manobras lúdicas. Ele também pode ser

treinado para não realizar certas ações: não fazer xixi dentro de casa, não sair correndo atrás do carteiro etc. Existe até mesmo um nível além, que poderia ser denominado como um "trabalho" por nós, seres humanos, como treinar o Scooby para farejar drogas ou servir de guia para pessoas cegas.

Tudo isso parece mágico, não? E realmente é, mas a palavra-chave aqui é "condicionamento". O treinamento de um cão geralmente envolve o **reforço positivo ou negativo** para solidificar comportamentos desejados e se livrar dos indesejados. Por exemplo, imagine que você e o Scooby estão passeando em um parque e você decide jogar uma bolinha de borracha para longe. O Scooby, curioso pela sua atitude, sai correndo atrás da bolinha, a pega com a boca e a traz de volta para você. Por essa atitude, você decide recompensá-lo com um petisco. Ele, então, pensa: "*Que legal, eu trouxe a bolinha e ele me deu um petisco!*" (perdão pela antropomorfização) e passa a repetir sempre essa ação de trazer a bolinha de volta, esperando pela recompensa. Isso seria o **reforço positivo** mencionado acima.

Já um exemplo de **reforço negativo** pode ser o ato de puxar o Scooby pela coleira quando ele se afasta muito de você, o que, na visão dele, causa certo desconforto e deve ser evitado no futuro. Porém, é claro, o Scooby poderia ter optado por não ir atrás da bolinha e, assim, não teria recebido o petisco. Dessa forma, esse método de aprendizado é um processo de tentativa e erro por parte do Scooby, a fim de otimizar suas recompensas e evitar punições.

Figura 2.2: Um cachorro em um parque correndo atrás de uma bola azul (imagem gerada pela IA Midjourney)

Porém, esse é só um dos tipos possíveis de aprendizado. Quando somos crianças, como a Paulinha, de três anos de idade, é comum termos adultos ao nosso redor para nos **supervisionar**. Naturalmente, eles vão nos **ensinando** diversas coisas, por exemplo, que "*aquela coisa arredondada, vermelha, que tem um cabinho no topo e está na fruteira*" se chama "maçã". Alguns dias depois, no supermercado, a Paulinha verá novamente aquele objeto junto a diversos outros, e sua mãe lhe dirá que aquilo é uma "maçã" também.

Dessa forma, depois de ver um determinado número de maçãs em diferentes contextos (em casa, na feira, no supermercado) e até mesmo em diferentes cores (vermelha, verde), nós e a Paulinha nos tornamos capazes de **generalizar** e **associar** esse tipo de objeto/imagem ao conceito de "maçã" que nos foi apresentado, e obtemos a habilidade de reconhecer "*objetos/imagens do tipo maçã*" mesmo em novos contextos (presa a uma árvore, desenhada em um livro ou no meio de cachos de banana, por exemplo).

Figura 2.3: Uma mãe e uma filha na cozinha com maçãs em uma vasilha sobre a mesa (imagem gerada pela IA Midjourney)

Um outro tipo de aprendizado, ainda falando sobre a infância, pode ser exemplificado pelo Pedrinho. O Pedrinho tem dois anos de idade e ganhou dos pais aquele brinquedo de plástico que contém diferentes formas geométricas (círculo, quadrado, triângulo etc.) que devem ser encaixadas cada uma em seu buraco correspondente. Os pais não lhe dizem nada, e ele começa a brincar.

Inicialmente, ele tenta inserir as formas geométricas em buracos aleatórios e, assim, começa a perceber que algumas formas se encaixam perfeitamente em alguns buracos, enquanto outras não. Ele repara, então, que as formas geométricas que se encaixam perfeitamente têm exatamente o mesmo formato dos buracos nos quais elas caem. Ou seja, mesmo sem nenhuma informação inicial, Pedrinho foi capaz de **encontrar os padrões** que faziam mais sentido e de fazer as combinações pertinentes.

Figura 2.4: Uma criança brincando com uma caixa que contém buracos e formas geométricas (imagem gerada pela IA Midjourney)

Esses princípios de recompensa, generalização e descoberta de padrões que vimos anteriormente formam a base do aprendizado de muitas inteligências artificiais. Assim como acompanhamos os processos de aprendizado do Scooby, da Paulinha e do Pedrinho, agora é hora de mergulharmos na mente de uma IA para desvendar os segredos por trás de suas habilidades cognitivas.

2.3 Machine Learning, o Aprendizado de Máquina

Bom, começamos a chegar às partes interessantes e mais técnicas. Uma definição possível de *Machine Learning* ou *Aprendizado de Máquina* é "*a programação de computadores para otimizar um critério de desempenho usando dados de exemplo ou experiências anteriores*"[1] .

Se preferir uma definição menos técnica, é como se o computador fosse um aspirante a chef tentando criar uma nova receita de sopa deliciosa. Ele começa com um conjunto básico de ingredientes (os dados de exemplo) e, talvez, uma receita aproximada e, a cada vez que tenta prepará-la, faz algumas alterações (otimiza o desempenho) com base no que funcionou ou não funcionou anteriormente (experiências anteriores). Com o tempo, o chef acaba criando uma sopa incrivelmente apetitosa e, o mais impressionante, sem necessariamente precisar de ninguém explicando explicitamente como fazê-lo. Ele consegue aprender e se aperfeiçoar por conta própria.

Nesta seção, veremos algumas das técnicas mais usadas na academia e no dia a dia do mercado de trabalho para desenvolver sistemas de *Aprendizado de Máquina*. Mas peço muita calma nessa hora. Se você, caro(a) leitor(a), não vem de um background de programação ou TI (ou, em muitos casos, até mesmo se vier), eu entendo que essa expressão "*Machine Learning*" possa assustar um pouco, mas acalme-se, e eu lhe garanto que os conceitos que veremos nesta seção são muito mais palatáveis do que possam parecer à primeira vista. O Scooby, a Paulinha e o Pedrinho serão prova disso.

Aprendizado Supervisionado

Conceito:

O *Aprendizado Supervisionado* é um tipo de *Aprendizado de Máquina* que consiste em treinar um sistema para que, ao receber um determinado dado de entrada, ele saiba apontar a saída correta ou mais provável. Isso é feito com base em um conjunto de dados rotulados usados em seu treinamento (aqui, a palavra "rotulado" indica que a saída esperada correspondente a essa entrada já é conhecida).

Esse tipo de aprendizado basicamente corresponde ao que vimos no caso da Paulinha na seção anterior. Sua mãe lhe mostrava "*aquela*

coisa arredondada, vermelha, que tem um cabinho no topo e está na fruteira" e rotulava esse objeto com a saída esperada: "*maçã*". Esse "treinamento" da Paulinha (que equivale ao modelo de IA, neste caso) prosseguiu com uma certa quantidade de entradas, em que sua mãe lhe mostrava essa fruta em diferentes locais ou imagens dela em um livro, e lhe oferecia o rótulo/saída esperada ("*Isso é uma maçã.*"). Imagine que, um dia, as duas estão passeando pela rua e veem essa fruta jogada ao lado de uma lixeira. Esse é um contexto novo, nunca visto pela Paulinha, e sua mãe lhe pergunta: "*O que é isso?*". Se o treinamento tiver sido bem-sucedido, a Paulinha (ou o modelo de IA) apontará corretamente que "*Isso é uma maçã*".

Esse também é um método equivalente a como aprendemos na escola ou com tutoriais online. Imagine que você está aprendendo a programar em *JavaScript* seguindo um curso da empresa Alura. Os instrutores lhe apresentam diversos conceitos novos, como variáveis, estruturas condicionais ou *loops* (que, nesse caso, equivalem às entradas) e os rotulam, explicando o que são e dando exemplos de uso na prática (que equivalem às saídas corretas). Você, então, depois de terminar o curso, tentará generalizar os conhecimentos adquiridos durante o treinamento para criar novas aplicações, diferentes daquelas vistas durante o curso. Se conseguir, isso significará que o modelo (você, nesse caso) foi bem treinado.

Devido a essas características, o *Aprendizado Supervisionado* é muito usado para resolver problemas de **classificação**, como no caso de rotular que "nesta imagem há uma maçã", "nesta imagem há uma laranja"; e problemas de **regressão**, onde queremos calcular um valor numérico de saída com base em variáveis de entradas, como prever o preço ideal de venda de uma propriedade com base em sua localização, número de quartos e no preço de outras propriedades similares na mesma zona.

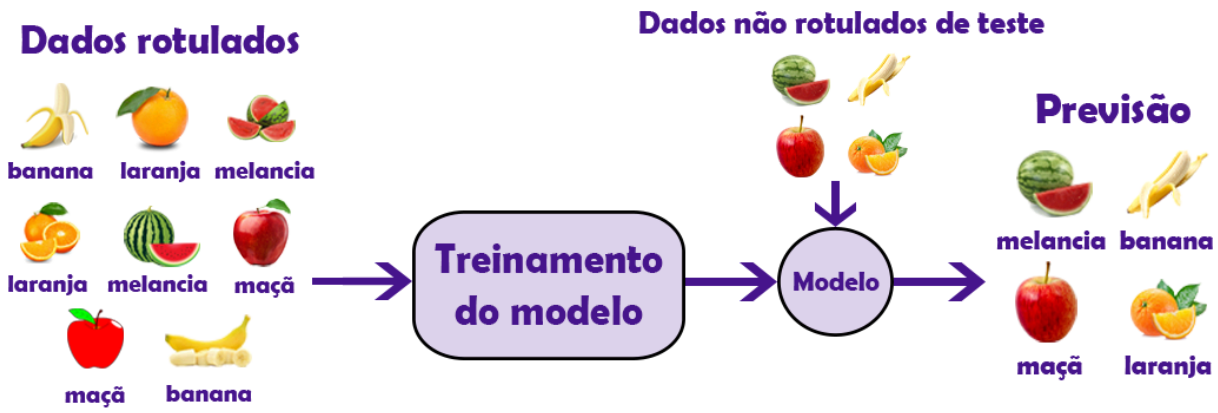

Figura 2.5: Fluxograma exemplificado de Aprendizado Supervisionado

Passo a passo na prática:

- **Definição e preparação dos dados rotulados**: o primeiro passo no *Aprendizado Supervisionado* é coletar e preparar um conjunto de dados que consista em pares que associem entradas a suas saídas correspondentes (rótulos). No caso de um sistema bancário, as entradas poderiam ser, por exemplo, dados de clientes que pediram empréstimos no passado; e as saídas, a informação sobre se estes clientes pagaram os empréstimos em dia ou se estão inadimplentes.
- **Separação dos dados em dois conjuntos**: o conjunto de dados inicial é embaralhado aleatoriamente e depois separado em dois novos — os dados que serão usados no treinamento no sistema e os dados que serão usados para testar sua eficácia. Geralmente, em torno de 80% dos dados iniciais são usados no treinamento e os 20% restantes são colocados de lado para a validação futura. Claro, essa divisão pode ser feita de outras maneiras (uma muito comum é com três conjuntos: ~70% para o treinamento, ~15% para a validação e 15% mantidos completamente de lado e usados apenas uma vez para o teste final de desempenho), mas o conjunto de treinamento será sempre maior que o de validação.

- **Escolha do algoritmo de aprendizado**: um algoritmo nada mais é que uma sequência de passos, como uma receita de bolo. Para *Aprendizado Supervisionado*, existem diversos algoritmos ou modelos que podem ser escolhidos, sendo que alguns são mais adequados para imagens, outros para análise de textos, outros para dados numéricos etc. Citarei apenas os nomes de alguns deles aqui, para caso você queira buscar saber mais sobre cada um deles: regressão linear[2] , regressão logística[3] , árvores de decisão[4] , florestas aleatórias[5] , máquinas de vetores de suporte (SVM)[6] , K-Nearest Neighbors (KNN) [7] , Naive Bayes[8] , Gradient Boosting (XGBoost e outros) [9] , AdaBoost[10] e Redes Neurais Artificiais (para as quais teremos uma seção dedicada).

- **Treinamento**: essa é a fase na qual usamos nosso conjunto de dados de treinamento (aqueles 80% do conjunto inicial total) para gradualmente melhorar a capacidade de predição do nosso modelo. O conjunto de dados de treinamento é usado como entrada do algoritmo de aprendizado selecionado, e o treinamento é realizado por meio de iterações, também conhecidas como "épocas". Cada iteração ou época representa uma passagem completa do algoritmo por todo o conjunto de dados de treinamento. Durante cada época, o modelo recebe as entradas do conjunto de treinamento e faz previsões com base em seus parâmetros (pesos e vieses) atuais. Na primeira época, começamos com valores aleatórios para os parâmetros e, como você pode imaginar, as previsões são muito ruins. Mas isso serve para calcular a função de perda, calibrar os valores e tentar otimizar as previsões.

  Uma **função de perda** (*loss function*)[11] é uma função matemática usada para medir o quão bem o modelo está se saindo durante o treinamento. Ela calcula a diferença entre as previsões do modelo e as saídas reais esperadas dos dados de treinamento. Além disso, também é usado um algoritmo de

otimização para atualizar os parâmetros do algoritmo de treinamento de acordo com a função de perda calculada. O objetivo é ajustar os parâmetros (pesos e vieses) de forma que a função de perda seja reduzida, o que significa que o erro está menor e as previsões, melhores.

Esse processo é repetido por várias "épocas", cuja quantidade é definida por quem criou o treinamento.

- **Validação**: após o treinamento, o desempenho do modelo é testado usando o conjunto de dados de validação (os 20% que foram separados lá no começo, lembra?), que contém exemplos não vistos pelo modelo durante o treinamento. Essa avaliação é importante para verificar se o modelo está generalizando bem para novos dados e se não está superajustado (*overfitting*) aos dados de treinamento.

  Usando nosso exemplo de um sistema bancário, nós rodaríamos o modelo treinado para os 20% de dados de testes e, como nós temos as saídas corretas para se esses clientes pagaram seus empréstimos ou estão inadimplentes, torna-se fácil verificar a taxa de precisão das previsões desse modelo.

- **Previsão**: uma vez que o modelo tenha sido treinado e avaliado adequadamente, ele estará pronto para fazer previsões em novos dados de entrada, para os quais não temos saídas ainda. Nosso hipotético modelo bancário, por exemplo, poderia tentar prever, com base nos padrões de clientes aprendidos durante o treinamento, se um novo cliente que está pedindo um empréstimo hoje provavelmente o pagará em dia ou se será inadimplente, e a qualidade desse modelo será medida pela sua taxa de acertos nessas previsões.

- **Otimização**: o modelo pode ser retreinado com mais dados caso se deseje melhorar sua taxa de acertos de previsão.

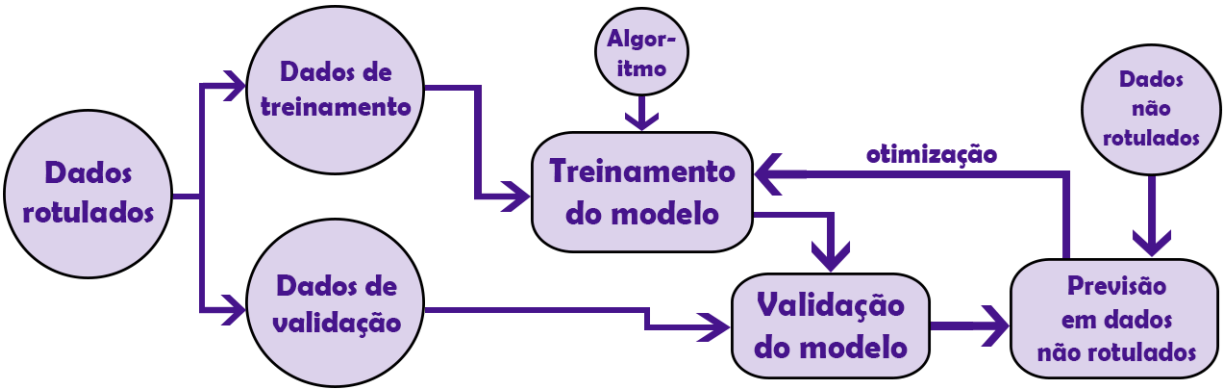

Figura 2.6: Fluxograma detalhado de Aprendizado Supervisionado

Vantagens do Aprendizado Supervisionado:

- Excelente para sistemas de classificação ou regressão (quando precisamos de um valor numérico);
- Fácil de compreender seu funcionamento interno quando comparado com outras técnicas;
- Maior controle sobre a definição das classes possíveis em problemas de classificação;
- Para muitos modelos, como a regressão linear ou redes neurais, o resultado do treinamento é uma função matemática. Isso significa que, após o treino, não é necessário manter os dados originais em memória para fazer novas previsões, apenas os parâmetros do modelo.

Desvantagens do Aprendizado Supervisionado:

- Necessidade de que os dados de treinamento estejam rotulados e sejam de qualidade. Conjuntos de dados de treinamento em pouca quantidade ou inconsistentes podem resultar em modelos ruins;
- A preparação dos dados de forma manual pode ser cara e demorada;
- Pode haver "sobreajuste" (*overfitting*)[12] caso o modelo seja treinado por muito tempo em uma amostra de dados, caso a amostra seja pequena ou quando haja ruído nos dados, e isso pode resultar em um modelo que se ajusta muito bem aos

dados de treinamento, mas não é capaz de generalizar bem para dados novos;
- Pode ter dificuldades em problemas nos quais não haja dados rotulados em quantidade ou nos quais as características relevantes não estejam claras;
- Pode exigir muitos recursos computacionais para o treinamento, especialmente para grandes conjuntos de dados.

Aplicações e casos de uso reais:

Como você pode imaginar, sistemas de IA que aprendem através do *Aprendizado Supervisionado* são extremamente comuns no seu dia a dia.

No âmbito financeiro, além do exemplo anterior sobre autorização de empréstimos, essa técnica também é muito usada para tentar prever fraudes em transações de cartão de crédito[13] e na identificação de padrões suspeitos de uso de planos de saúde.[14] Porém, além de tentar mitigar perdas, o *Aprendizado Supervisionado* também é extensivamente utilizado com o intuito de impulsionar ganhos, como em sistemas de previsão para estratégias de investimento em operações no mercado de ações[15] e para empresas do mercado imobiliário.[16]

Sistemas de recomendação são um outro exemplo clássico do uso de *Aprendizado Supervisionado*. Gigantes do comércio virtual, como a *Amazon* e a *Alibaba*, sites de streaming, como a *Netflix*[17] , e redes sociais, incluindo o *TikTok* e o *Instagram*, têm implementado e aperfeiçoado técnicas de *Aprendizado Supervisionado* ao longo dos anos em seus sistemas de recomendação, a fim de sugerir conteúdos e produtos[18] personalizados ao gosto dos seus usuários ou prever a probabilidade de um usuário interagir com um item (clicar, comprar, curtir).

Por fim, podemos citar empresas de notícias, como jornais e revistas online, que podem implementar um sistema de recomendação do estilo "*Leia mais artigos*". Esses sistemas se baseiam nas

preferências individuais de cada leitor, direcionando-os para ler mais conteúdo de seu interesse dentro de categorias específicas que lhes agradam.[19]

Porém, como vimos no tópico de "Desvantagens", o *Aprendizado Supervisionado* não é uma solução única e nem é ideal para todos os tipos de problema. Vamos ver alguns outros tipos de *Aprendizado de Máquina*.

Aprendizado Não Supervisionado

Conceito:

O *Aprendizado Não Supervisionado* é um tipo de *Aprendizado de Máquina* que consiste em tentar encontrar padrões "escondidos" nos dados e, assim, agrupá-los ou aprender possíveis relações entre eles. Diferentemente do *Aprendizado Supervisionado*, aqui, trabalhamos apenas com dados não rotulados, ou seja, não temos como indicar para o sistema qual é a saída esperada para os dados de entrada durante o treinamento.

Esse tipo de aprendizado corresponde ao que vimos no caso do Pedrinho na seção 2.1. Seus pais lhe deram aquele brinquedo que contém peças de diferentes formas geométricas (círculo, quadrado, triângulo etc.), que devem ser encaixadas cada uma em seu buraco correspondente. Porém, eles não lhe ensinaram nada (ou, em termos de *Aprendizado de Máquina*, não *rotularam* os dados de entrada), apenas ofereceram o brinquedo ao Pedrinho e deixaram que ele próprio tentasse descobrir se aquelas peças em sua frente tinham alguma relação ou algum padrão.

No começo, obviamente, Pedrinho tem alguma dificuldade em entender como o brinquedo funciona e apenas tenta colocar as peças em buracos aleatórios. Porém, com o tempo, ele percebe que a peça triangular entra perfeitamente no buraco que tem, também, um formato triangular. Ele vê, então, que o mesmo é verdadeiro para a peça circular no buraco circular e para a peça quadrada no

buraco quadrado. Ele consegue, assim, completar o jogo, tendo encontrado padrões, similaridades e diferenças entre os dados iniciais (peças e buracos do brinquedo) que, inicialmente, não eram óbvios para si.

Algo similar ocorre quando nos mudamos para uma nova cidade onde não conhecemos ninguém. Inicialmente, tudo parece um enorme caos e a estrutura urbana não faz sentido para nós. Porém, com o tempo, começamos a perceber alguns padrões, como "*nessa rua tem muitos bares e baladas*", "*esse é um bairro residencial*", "*nessa zona existem ótimos restaurantes asiáticos*", "*esses são os ônibus que posso tomar para ir para casa*", e aquele caos inicial passa a ter alguma lógica.

Devido a essas características, o *Aprendizado Não Supervisionado* é principalmente útil para problemas de **agrupamento** (*clustering*, em inglês)[20] , onde queremos encontrar subgrupos dentro dos dados, como no caso de uma empresa criando segmentações de clientes com base em semelhanças de interesses ou demografia; problemas de **associação**, como para entender a correlação entre produtos que clientes compram juntos (pão e manteiga, por exemplo); e problemas de **detecção de anomalias**, em que poderia ser aplicado para um sistema de detecção de fraudes, por exemplo.

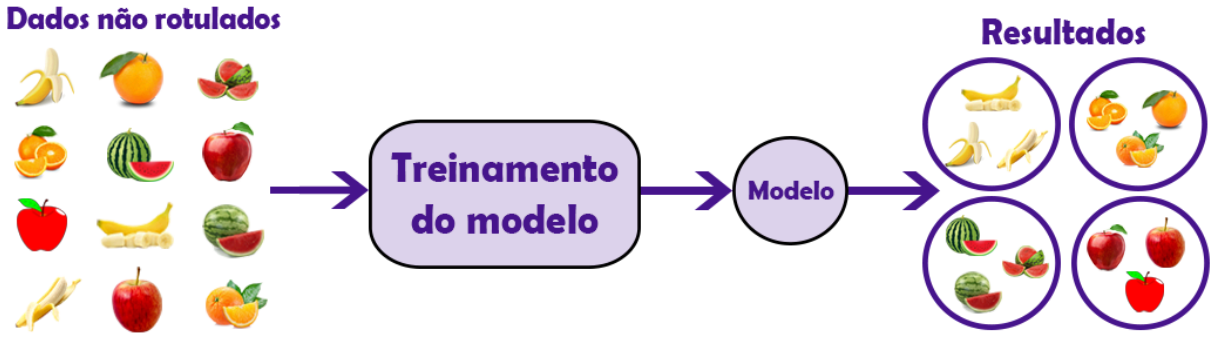

Figura 2.7: Fluxograma exemplificado de Aprendizado Não Supervisionado

Passo a passo na prática:

- **Definição e preparação dos dados**: o primeiro passo no *Aprendizado Não Supervisionado* é coletar e preparar um conjunto de dados que, diferentemente do *Aprendizado Supervisionado*, não precisam estar rotulados. Em um sistema de comércio virtual, por exemplo, poderiam ser os dados demográficos e de compras efetuadas dos clientes.

- **Escolha do algoritmo de aprendizado**: para *Aprendizado Não Supervisionado*, também existem diversos algoritmos ou modelos que podem ser escolhidos. Citarei apenas os nomes de alguns deles aqui, para caso você queira buscar saber mais sobre cada um deles: K-means[21] , Análise de Agrupamentos Hierárquicos (HCA)[22] , DBSCAN[23] , Maximização de Expectativa (EM)[24] , Análise de Componentes Principais (PCA)[25] , Decomposição em Valores Singulares (SVD)[26] , Modelo de Misturas Gaussianas (GMM)[27] , Mean Shift[28] , Apriori[29] , AutoEncoders[30] e Redes Neurais Artificiais (para as quais teremos uma seção dedicada).

- **Treinamento**: nesta fase, o algoritmo escolhido tentará encontrar padrões escondidos no conjunto de dados de entrada. Vamos supor que escolhamos o algoritmo *K-means* de agrupamento (*clustering*) para o nosso modelo. Nesse caso, o algoritmo começará selecionando "K" pontos iniciais aleatórios como "centroides" (por isso o nome "*K-means*"), que atuarão como os centros dos agrupamentos (*clusters*) — o valor de quantos centroides (quantos "K") teremos é predefinido por quem está rodando o treinamento.

  Cada ponto de dados no conjunto será inicialmente atribuído ao centroide mais próximo de si. Depois que todos os pontos forem atribuídos a um grupo, os centroides serão recalculados, com base na média de todos os pontos dentro de um *cluster*, e um novo centroide será criado em seu lugar. Esse processo será

repetido várias vezes, até que os centroides não mudem significativamente entre as iterações ou até que um número predeterminado de iterações seja alcançado.

No final, teremos grupos de pontos de dados que, segundo este algoritmo, são semelhantes entre si.

- **Avaliação dos resultados**: a avaliação dos resultados de um modelo de *Aprendizado Não Supervisionado* é, no geral, subjetiva, e sua interpretação pode ser um desafio, uma vez que não existem rótulos claros com os quais possamos comparar, como seria no *Aprendizado Supervisionado*. Lembre-se de que o *Aprendizado Não Supervisionado* não visa produzir respostas exatas, mas sim buscar padrões potenciais nos dados e oferecer *insights* para quem os analisa. No nosso cenário de agrupamento (*clustering*), por exemplo, poderíamos observar quais dados o algoritmo reuniu em cada agrupamento (*cluster*) para entender melhor o que eles têm em comum.

- **Otimização**: com base na avaliação dos resultados, o modelo pode ser retreinado. Isso pode ser feito ajustando os parâmetros do algoritmo, escolhendo um algoritmo diferente ou preparando o conjunto de dados de entrada de uma forma diferente.

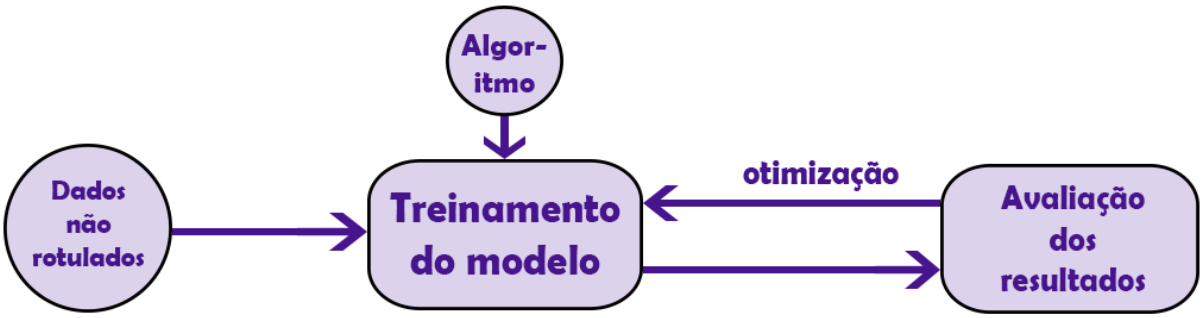

Figura 2.8: Fluxograma detalhado de Aprendizado Não Supervisionado

Vantagens do Aprendizado Não Supervisionado:

- Ideal para situações com dados não rotulados, que são, no geral, mais fáceis de obter;
- Acelera o processo de coleta e preparação dos dados, especialmente para grandes quantidades, por não exigir que sejam rotulados;
- Assim que o modelo agrupar os dados, podemos analisar os grupos formados e aplicar rótulos a eles;
- Pode revelar relações e padrões inesperados entre os dados, que seriam praticamente impossíveis para um ser humano perceber.

Desvantagens do Aprendizado Não Supervisionado:

- Como o modelo não sabe exatamente o que deve aprender, existe um alto risco de obter resultados irrelevantes ou imprecisos;
- Maior tempo necessário para avaliar e interpretar manualmente os resultados;
- Possível dificuldade na interpretação dos resultados;
- Maior dependência de uma boa qualidade dos dados de entrada, pois a existência de muitos dados errôneos ou irrelevantes pode levar a resultados enganosos.

Aplicações e casos de uso reais:

Tais quais sistemas de *Aprendizado Supervisionado*, sistemas de IA que aprendem através do *Aprendizado Não Supervisionado* também são facilmente encontrados no seu dia a dia. Páginas de busca ou de notícias, como o *Google News*, usam essa técnica para tentar agrupar notícias semelhantes em um mesmo segmento.[31]

Muitos sistemas de detecção de fraudes online usam *Aprendizado Não Supervisionado* para identificar comportamentos anormais ou atípicos que podem indicar atividades fraudulentas.[32] Por exemplo, um sistema pode sinalizar como suspeito se uma conta estiver fazendo compras excepcionalmente grandes ou a partir de locais incomuns. O mesmo é válido para fraudes de cartão de crédito.[33]

Como mencionado anteriormente, um dos usos de destaque dessa técnica é por empresas de comércio virtual, como a *Alibaba*[34] e a *Amazon*, que as utilizam para segmentar seus clientes em categorias com base nas coisas que eles compram e quanto gastam em geral, e, assim, poder definir quem é a sua *persona* alvo, encontrar padrões de vendas cruzadas e direcionar suas estratégias de marketing para grupos específicos.

Na biologia, o *Aprendizado Não Supervisionado* é utilizado no campo de design de proteínas[35] , que é o processo de criação de novas proteínas com propriedades específicas por meio da manipulação de sequências de aminoácidos.

Por fim, podemos citar novamente plataformas de streaming como a *Netflix*, que já afirmou mais de uma vez[17] [36] que utiliza em seus sistemas internos de recomendação e análise tanto modelos de *Aprendizado Supervisionado* quanto de *Aprendizado Não Supervisionado*.

Vendo essas aplicações, você já deve ter começado a notar que os mesmos sistemas e empresas podem usar diferentes técnicas de aprendizado em seus produtos, com diferentes finalidades.

Aprendizado Semi-Supervisionado

Conceito:

O *Aprendizado Semi-Supervisionado* pode ser considerado uma mescla entre o *Aprendizado Supervisionado* e o *Aprendizado Não Supervisionado* que vimos acima (por isso o nome *Semi-Supervisionado*). Esta é uma técnica de *Aprendizado de Máquina* que combina um pequeno número de dados rotulados com um conjunto de dados majoritariamente não rotulados durante o treinamento.

Usando o exemplo do Pedrinho da seção 2.1, imagine que, em vez de seus pais lhe darem o brinquedo das formas geométricas e

deixarem que ele próprio tentasse descobrir se aquelas peças têm alguma relação ou algum padrão com a caixa e os buracos, agora, seus pais lhe ensinassem que a peça triangular se encaixa perfeitamente no buraco de formato triangular, mas não falassem nada sobre as outras peças de formatos diferentes. Dessa forma, o Pedrinho já teria, então, uma base a partir da qual seguir para tentar classificar ou encontrar padrões entre o resto das peças.

Esse método de aprendizado é especialmente útil para casos em que temos uma grande quantidade de dados não rotulados e que o processo de rotulação seja muito dispendioso. Assim, podemos rotular apenas uma pequena parte dos dados, economizando tempo e, potencialmente, melhorando a performance de um modelo de *Aprendizado Supervisionado* puro, devido à possibilidade de utilizar grandes quantidades de dados não rotulados e encontrar novos padrões.[37]

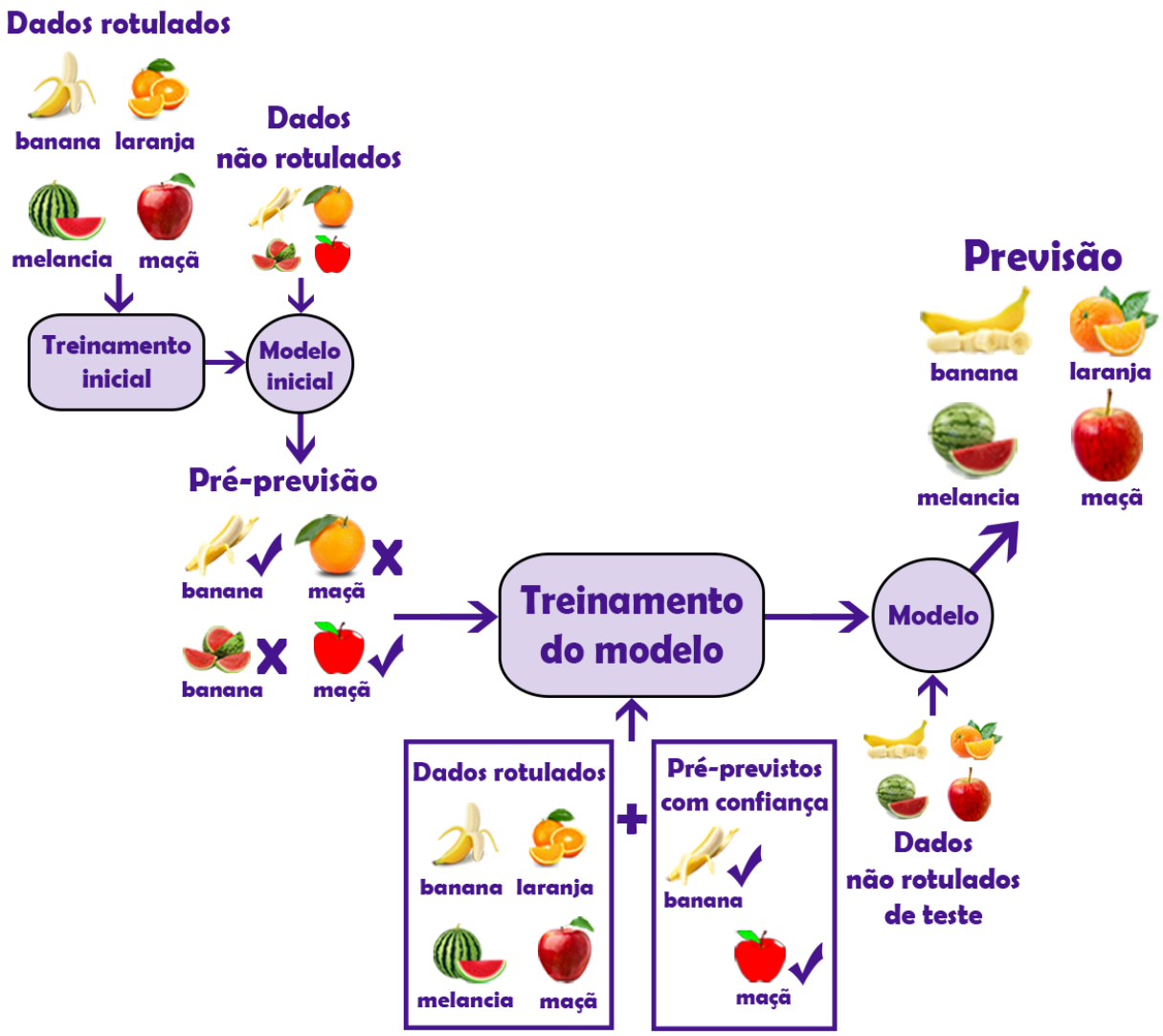

Figura 2.9: Fluxograma exemplificado de Aprendizado Semi-Supervisionado

Passo a passo na prática:

Há diversos algoritmos que podem ser aplicados para treinar um modelo de *Aprendizado Semi-Supervisionado*, como os baseados em grafos, Propagação de Rótulos (*Label Propagation*)[38] , Autotreinamento[39] , Cotreinamento[40] , Co-Forest[41] e Redes Neurais Artificiais. Usaremos o algoritmo de **Autotreinamento** para o exemplo a seguir.

- **Definição e preparação dos dados**: assim como no *Aprendizado Supervisionado*, será preciso coletar e preparar um

conjunto de dados rotulados, ou seja, que consista em pares que associem entradas a suas saídas correspondentes (rótulos). Existe também um conjunto majoritário de dados não rotulados, que não será usado neste primeiro momento.

- **Separação dos dados em dois conjuntos**: também como no *Aprendizado Supervisionado*, o conjunto de dados inicial é separado em dois novos — os dados que serão usados no treinamento no sistema e os dados que serão usados para testar sua eficácia — em torno de 80% para o pré-treinamento e 20% para a validação. Claro, essa divisão pode ser feita de outras maneiras, mas o conjunto de treinamento será sempre maior que o de validação.

- **Treinamento inicial**: usando um algoritmo de *Aprendizado Supervisionado* escolhido, o modelo é pré-treinado apenas com o conjunto de 80% de dados rotulados.

- **Previsão nos dados não rotulados**: adicionamos nesta etapa os nossos dados não rotulados, que são o maior conjunto de dados, e usamos o modelo pré-treinado com nossos poucos dados rotulados para fazer previsões sobre os dados não rotulados. Neste ponto, essas previsões ainda são bastante incertas, por isso, chamamos os rótulos de "pseudo-rótulos".

- **Seleção das previsões de maior confiança**: selecionamos os dados pseudo-rotulados com maior confiança, e eles serão usados para treinar o modelo.

- **Retreinamento**: nesta fase, retreinamos o modelo utilizando o nosso conjunto inicial de dados rotulados e também o conjunto de dados pseudo-rotulados com maior confiança do passo anterior.

- **Validação**: após o retreinamento, o desempenho do modelo é testado usando o nosso conjunto de dados de validação (os 20% restantes de dados rotulados do início).

- **Otimização**: o modelo pode ser retreinado com mais dados caso se deseje melhorar sua taxa de acertos de previsão.

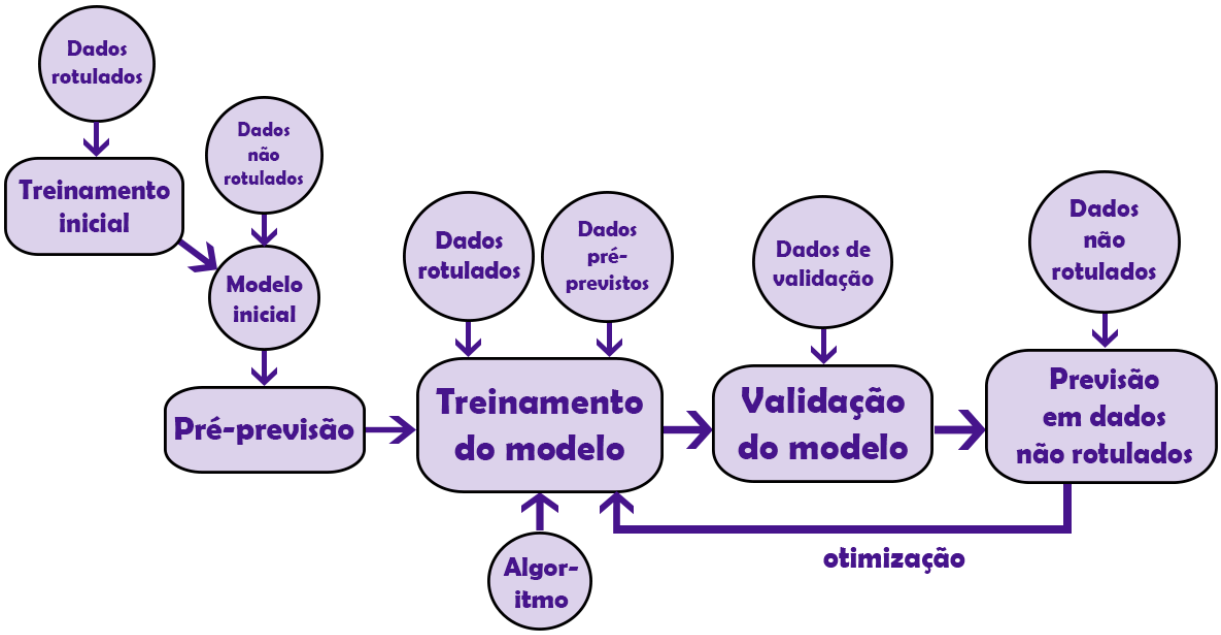

Figura 2.10: Fluxograma detalhado de Aprendizado Semi-Supervisionado

Vantagens do Aprendizado Semi-Supervisionado:

- Poderoso quando temos muitos dados, mas poucos deles rotulados;
- Economiza consideravelmente tempo e gastos na preparação e rotulagem dos dados, por não exigir que todos sejam rotulados;
- Potencialmente, maior capacidade de generalização e reconhecimento de padrão mais sutis, por ter acesso a um grande número de dados não rotulados.

Desvantagens do Aprendizado Semi-Supervisionado:

- Maior complexidade de implementação do algoritmo;
- Mais sensível a dados não rotulados inconsistentes, o que pode prejudicar o desempenho do modelo;
- Nem todas as tarefas de *Aprendizado de Máquina* se beneficiam igualmente da inclusão de dados não rotulados, algumas podendo até ser prejudicadas.

Aplicações e casos de uso reais:

O exemplo clássico de *Aprendizado Semi-Supervisionado* é o sistema de marcação de pessoas em fotos em redes sociais como o *Facebook*. Imagine que, mês passado, você fez o upload de uma foto na qual aparecem você e sua amiga Luana, e você marcou o perfil dela nessa foto. Hoje, você saiu com ela e com outras pessoas, chegou em casa e está fazendo o upload de mais fotos com esse grupo de amigos. O modelo do Facebook consegue reconhecer que existe um rosto que aparece em 4 das 7 fotos que você subiu hoje, e como você já a marcou anteriormente em uma foto no mês passado (ou seja, criou um rótulo para aquele rosto), o modelo é capaz de marcá-la em todas as novas fotos automaticamente também.[42]

Podemos mencionar também a Netflix. Em seu sistema, poderíamos utilizar, por exemplo, as classificações dos usuários sobre filmes e séries assistidos como dados rotulados e os filmes e séries não avaliados por eles como dados não rotulados. Um algoritmo de *Aprendizado Supervisionado*, como a regressão logística ou a árvore de decisão, poderia ser usado para analisar os dados rotulados, identificar "*padrões e relacionamentos entre as preferências dos usuários*", e aplicar esses padrões aos dados não rotulados. Depois desse treinamento, o sistema seria capaz de "*fazer previsões e recomendações mais precisas e personalizadas, mesmo para filmes e séries não avaliados anteriormente.*"[43]

Outro uso muito comum é na análise de imagens médicas. No contexto da radiologia, um profissional da área poderia atuar juntamente a uma pessoa cientista de dados na preparação dos dados, analisando e rotulando um pequeno conjunto de imagens de exames para verificar se existe, por exemplo, algum tumor nos testes analisados. Porém, um modelo de *Aprendizado Supervisionado* padrão precisaria de uma enorme quantidade de dados de entrada para ter uma boa precisão. Agora, imagine se esse profissional tivesse que fazer isso manualmente para todos os

exames que serão os dados de entrada do modelo? Isso seria extremamente dispendioso (e é o padrão em muitos dos casos). No entanto, com o *Aprendizado Semi-Supervisionado*, usar um pequeno conjunto de dados rotulados para pré-treinar o modelo pode "*melhorar sua precisão em comparação com um modelo totalmente Não Supervisionado.*"[44]

Por fim, ele pode ser usado em modelos de reconhecimento de fala, em que a rotulação dos dados (transcrição de arquivos de áudio) é muito dispendiosa. Porém, é possível usar um pequeno conjunto rotulado com um grande conjunto de dados não rotulados para treinar o modelo com sucesso.[45]

Aprendizado Autossupervisionado

Conceito:

O *Aprendizado Autossupervisionado* (*Self-Supervised Learning — SSL*) é uma técnica mais recente, e que ganhou muita força nos últimos anos. É a técnica principal por trás do pré-treinamento de muitos dos modelos mais poderosos da atualidade, como o *ChatGPT*, o *BERT*, o *CLIP* e vários modelos de visão computacional ou de fala.

Apesar dos nomes parecidos, ele é bem diferentemente do *Aprendizado Semi-Supervisionado*. No *Aprendizado Autossupervisionado* não há nenhum rótulo humano, nem mesmo em pequena quantidade. Embora seja frequentemente considerado um subconjunto do *Aprendizado Não Supervisionado* por não precisar de rótulos criados por humanos, ele se destaca por uma abordagem engenhosa: ele cria seus próprios rótulos a partir dos dados de entrada, que originalmente não são rotulados. Em vez de depender de um humano para dizer o que é certo ou errado, o modelo aprende sobre a estrutura inerente e o contexto dos dados ao resolver um quebra-cabeça que ele mesmo criou.

Usando novamente o Pedrinho como exemplo, vamos imaginar que ele, já um pouco mais velho, esteja tentando aprender a ler sozinho. Seus pais lhe deram vários livros, mas sem ler para ele. O Pedrinho pega uma frase de um livro, como "*O gato está ____ em cima do telhado*.", e cobre uma palavra com o dedo. A tarefa que ele mesmo se impõe (o que chamamos de "tarefa pretexto") é adivinhar qual palavra está faltando. Para fazer isso, ele precisa entender o contexto: o que um gato faria "em cima do telhado"? Ele provavelmente não estaria "nadando" ou "voando". Talvez ele esteja "dormindo", "andando" ou "pulando". Ao remover o dedo e verificar a resposta correta ("dormindo"), o Pedrinho se "autocorrige". Mesmo que não haja ninguém dizendo qual é a resposta "certa", ele aprende com base nas estruturas e padrões da linguagem que já viu antes.

Repetindo esse processo milhares de vezes com diferentes frases, ele não está apenas aprendendo a preencher lacunas, mas também aprendendo gramática, semântica e como as palavras se relacionam para formar ideias coerentes, tudo isso sem que um professor ou os pais lhe forneçam as respostas diretamente.

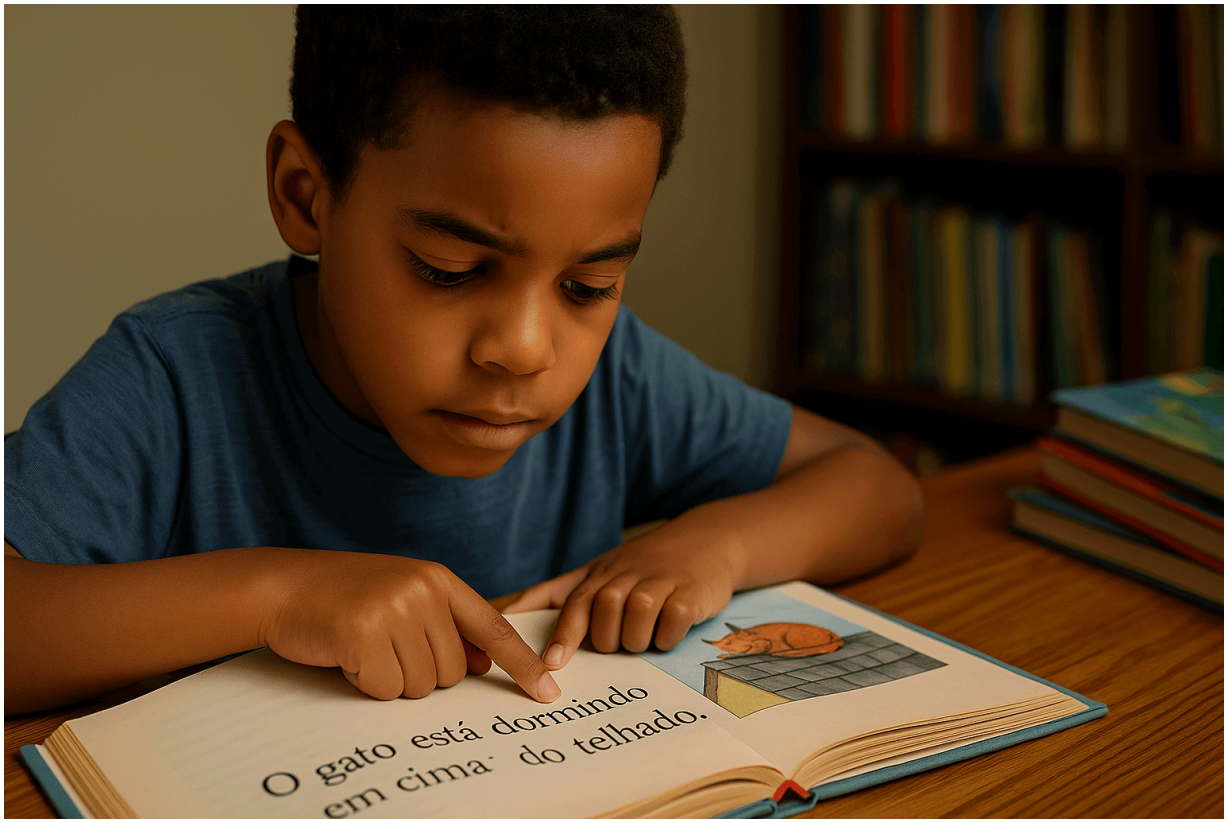

Figura 2.11: Uma criança mais velha lendo uma frase em um livro (imagem gerada pela IA ChatGPT-4o)

É essa capacidade de gerar a própria supervisão que permite aos modelos de Aprendizado Autossupervisionado aprenderem representações ricas e profundas a partir de enormes quantidades de dados brutos (como todos os textos da internet), formando a base para os modernos modelos de linguagem, como o *ChatGPT*.

Passo a passo na prática:

Embora existam diversas abordagens, a lógica geral do Aprendizado Autossupervisionado costuma seguir estas etapas:

- **Preparação dos dados brutos**: em vez de dados rotulados por humanos, são usados dados brutos, como textos de sites, imagens da internet, gravações de áudio, entre outros. Para um

modelo de linguagem, isso pode ser uma coleção gigantesca de textos da Wikipédia, livros, artigos de notícias e sites. Para um modelo de visão, podem ser milhões de imagens da internet.

- **Criação automática de tarefas pretexto**: aqui está o "truque" do método. O próprio sistema define desafios a partir dos dados. Alguns exemplos de tarefas pretexto são:
    - em texto, esconder uma palavra da frase e tentar prever qual era, ou então, dado um trecho de texto, treinar o modelo para prever a próxima palavra;
    - em imagens, o modelo pode receber um pedaço de uma imagem e ter que prever um pedaço vizinho, ou receber uma imagem em preto e branco e ter que adivinhar suas cores originais;
    - em áudio: remover uma parte e pedir para o modelo completá-la.
- **Pré-treinamento do modelo**: um modelo de rede neural grande (como uma arquitetura *Transformer*) é treinado para resolver as tarefas pretexto, comparando sua resposta com o "rótulo" que ele mesmo pode verificar (por exemplo, a palavra escondida originalmente). O objetivo aqui não é se tornar perfeito em adivinhar palavras ou colorir imagens, mas sim forçar o modelo a aprender as características e padrões fundamentais dos dados para poder resolver o problema. É nessa fase que o modelo desenvolve uma "compreensão" do idioma ou do conteúdo visual.

- **Ajustes e *fine-tuning* (opcional)**: após o pré-treinamento autossupervisionado, teremos um "modelo de base", que é muito bom em entender dados. Esse modelo pode passar agora por um *fine-tuning* (um treinamento específico), desta vez supervisionado, com uma pequena quantidade de dados rotulados, caso se deseje especializá-lo em uma tarefa específica, como tradução, geração de código, classificação de sentimentos etc. Como o modelo já "entende" a linguagem, ele

aprende a nova tarefa muito mais rápido e com mais precisão do que um modelo treinado do zero.

Vantagens do Aprendizado Autossupervisionado:

- Não precisa de grandes quantidades de dados rotulados manualmente, que é um processo caro, demorado e propenso a erros;
- Permite aproveitar enormes volumes de dados brutos não rotulados disponíveis publicamente, como toda a internet, textos ou imagens de redes sociais;
- Cria modelos com alto grau de generalização, capazes de aprender representações ricas dos dados;
- É a tecnologia fundamental que tornou possíveis os LLMs (Large Language Models) e outros "modelos de fundação" de grande escala.

Desvantagens do Aprendizado Autossupervisionado:

- Requer alto poder computacional para treinar em grandes volumes de dados;
- A concepção equivocada de uma "tarefa pretexto" pode levar o modelo a aprender características inúteis ou irrelevantes;
- A qualidade do aprendizado depende fortemente da qualidade e diversidade dos dados brutos;
- A fase de pré-treinamento exige enormes recursos computacionais (milhares de GPUs ou TPUs rodando por semanas ou meses) e, consequentemente, tem um alto custo financeiro e ambiental.

Aplicações e casos de uso reais:

O *Aprendizado Autossupervisionado* é, sem dúvida, um dos maiores motores da revolução da IA que vivemos hoje. Sua aplicação mais famosa está nos *Large Language Models*. O modelo *BERT*, do *Google*, popularizou a técnica de mascaramento de palavras para pré-treinar um sistema que alcançou um entendimento de contexto sem precedentes. As séries de modelos *GPT*, da *OpenAI* e *Gemini* do

Google utilizam uma abordagem similar. Eles são pré-treinados em vastas quantidades de texto e código da internet sem rótulos explícitos e, ao aprender a prever a próxima palavra ou preencher lacunas em frases, eles desenvolvem uma compreensão profunda de gramática, raciocínio e conhecimento geral.

A geração de imagens, vídeos e áudio a partir de texto é outra aplicação direta. Modelos como *Midjourney*, *Veo* e *Sora* aprendem a associação entre palavras e padrões visuais analisando bilhões de pares de imagem-texto da internet. Essa "autossupervisão" permite que eles "entendam" conceitos abstratos e gerem conteúdo visualmente coerente para prompts como "*um astronauta montado em um cavalo realista na lua*" ou "*um vídeo do Will Smith comendo macarronada*".

O Reconhecimento de Fala também se beneficia enormemente. Os modelos conseguem aprender representações fundamentais da fala humana a partir de milhares de horas de áudio bruto, sem transcrições. Depois, eles podem ser rapidamente ajustados com poucos dados rotulados para criar sistemas de transcrição de alta precisão para assistentes de voz e outros serviços.

No setor da saúde, ele está revolucionando a medicina diagnóstica. Por exemplo, pode-se pré-treinar um modelo em um enorme banco de imagens médicas não rotuladas de forma autossupervisionada para aprender a reconhecer padrões visuais gerais, e então fazer um *fine-tuning* (um treinamento específico) supervisionado com um conjunto relativamente pequeno de imagens rotuladas (por especialistas indicando, por exemplo, presença de um tumor). Estudos mostram que essa abordagem pode identificar sinais precoces de doenças como câncer de mama em mamografias ou retinopatia diabética em exames de retina com altíssima precisão, muitas vezes comparável ou melhor do que especialistas humanos.

Aprendizado por Reforço

Conceito:

O *Aprendizado por Reforço* é um tipo de *Aprendizado de Máquina* através do qual um **agente** (aqui, o modelo de IA) aprende a tomar decisões interagindo com seu **ambiente** (preste atenção nessas duas palavras).

Ela funciona com base em um sistema de recompensas, onde o agente começa sem conhecimento sobre o *ambiente* em que está funcionando e vai aprendendo através da experimentação e interação contínua com ele. Através da tentativa e erro, o agente começa tomando decisões aleatórias, mas vai aprendendo com o passado, repetindo decisões que antes resultaram em recompensas a fim de maximizar os *reforços positivos*, e evitando ações que tenham resultado em punições, a fim de minimizar os *reforços negativos*.

Como você já deve estar imaginando, esse tipo de aprendizado corresponde ao caso do nosso cãozinho Scooby na seção 2.1. Quando você joga a bolinha de borracha para longe (situação do ambiente), são apresentadas ao Scooby (o agente) algumas opções de *ação*, como "ir correndo atrás dela e trazê-la de volta" ou "não fazer nada". Caso ele decida trazê-la de volta, você lhe dá um petisco (reforço positivo ou recompensa) e, com isso, começa a condicioná-lo a tomar a mesma atitude novamente no futuro, a fim de obter mais recompensas. Caso ele não vá atrás da bola ou corra para o lado oposto, não receberá nenhuma recompensa.

Porém, se o Scooby tomar uma atitude indesejada, como se afastar muito de você ou tentar morder o carteiro, você pode, por exemplo, puxá-lo pela coleira, o que causará certo desconforto (punição ou reforço negativo) e, idealmente, o condicionará a evitar tomar essa mesma atitude novamente, a fim de evitar essa punição.

Este tipo de aprendizado é útil em situações nas quais o modelo de IA precisa operar em ambientes complexos e incertos, tendo que

tomar uma sequência de decisões cujas consequências talvez só se tornem claras muito mais tarde.

Imagine um ratinho gerado por uma IA de *Aprendizado por Reforço* de nome Eustáquio (nosso agente). Ele foi colocado em um labirinto (ambiente) com o objetivo de encontrar a saída (como geralmente é o caso quando seres são colocados dentro de labirintos).

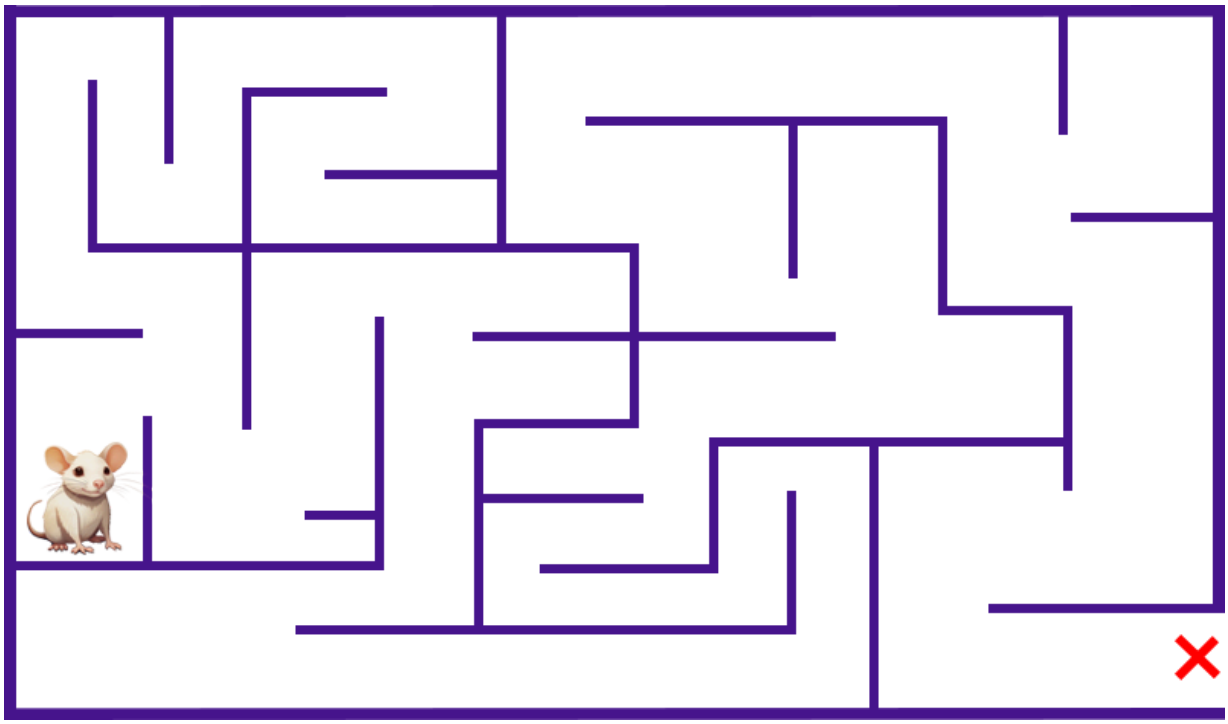

Figura 2.12: Exemplo de Aprendizado por Reforço - Rato em labirinto - Estado inicial

O Eustáquio, podendo escolher ir para cima, baixo, direita ou esquerda, começa fazendo movimentos aleatórios. Sempre que ele fizer uma ação que o aproxime da saída, podemos lhe dar uma recompensa, como um pedaço de queijo. Se a sua ação o afastar da saída ou o direcionar a um beco sem saída, podemos optar por não lhe dar nada ou lhe dar uma pequena punição (dependendo do algoritmo escolhido).

Conforme experimenta as recompensas e penalidades de suas ações, o Eustáquio começará a descobrir que seguir certos caminhos ou tomar certas ações resulta em uma recompensa total maior — preste atenção na expressão "recompensa total", pois o agente poderá, por exemplo, ter que caminhar por um tempo em uma direção contrária à da saída, o que pode lhe ocasionar uma punição ou diminuição da recompensa momentânea, mas que, no final das contas, o fará chegar à saída de modo eficiente e maximizará a sua recompensa total.

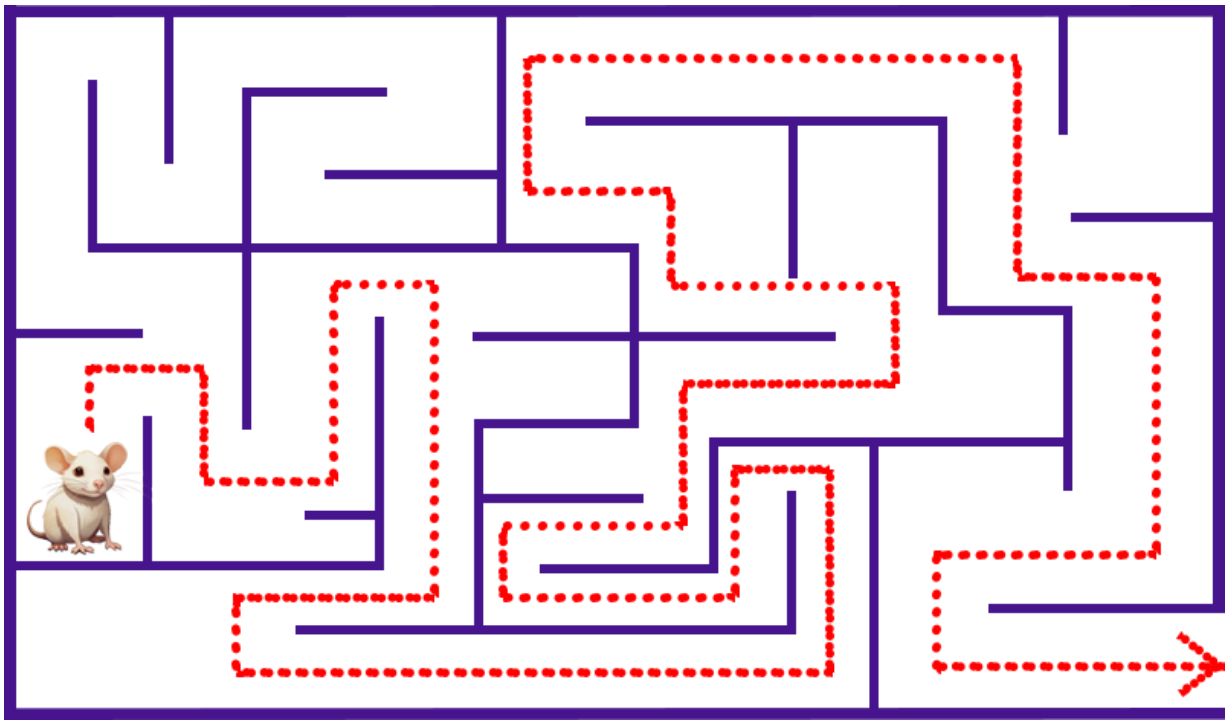

Figura 2.13: Exemplo de Aprendizado por Reforço - Rato em labirinto - Estado final

Passo a passo na prática:

Os algoritmos de *Aprendizado por Reforço* mantêm um equilíbrio entre a exploração (*exploration*) e o aproveitamento (*exploitation*). **Exploration** é o processo de tentar coisas diferentes aleatoriamente, a fim de verificar se recompensas maiores do que as obtidas anteriormente podem ser alcançadas. **Exploitation** é o processo de tomar uma ação mais "garantida", usando como base ações que obtiveram boas recompensas no passado.

Com isso em mente, temos:

- **Definição do problema e da política inicial**: começamos definindo o problema, o que envolve identificar o "ambiente" no qual o "agente" vai interagir, as "ações" que ele poderá executar e a "recompensa" que buscará maximizar. O "ambiente" é geralmente formulado como um *Processo de Decisão de Markov* (MDP)[46] que consiste em estados, ações, transições entre estados e recompensas. A "política" define a estratégia que o agente seguirá para tomar decisões, que pode ser determinística (uma ação específica para cada estado) ou estocástica (com base na probabilidade de escolher cada ação).

- **Interação com o ambiente**: o agente é colocado em um estado inicial dentro do ambiente. Ele observa o estado atual e, com base na política atual, escolhe uma ação para realizar no ambiente.

- **Transição e recompensa**: após o agente realizar a ação, o ambiente transita para um novo estado e o agente recebe uma recompensa ou uma punição.

- **Atualização da política**: o agente usa o *feedback* recebido (positivo ou negativo) e a nova observação de estado para ajustar sua política. Isso é feito usando um algoritmo de *Aprendizado por Reforço*, como o Q-learning[47] , Policy Gradient[48] ou SARSA[49] .

- **Treinamento iterativo**: o agente continua repetindo os passos acima, isto é, observando o novo estado do ambiente, escolhendo uma ação, recebendo uma recompensa (ou punição) e ajustando sua política, até que uma condição de parada seja atendida. Essa condição de parada pode ser uma quantidade máxima de tempo de treinamento, uma pontuação específica alcançada ou uma análise de que o agente não está melhorando o seu desempenho há um determinado número de iterações.

- **Avaliação**: ao longo do processo de treinamento, é importante monitorar o desempenho do agente, avaliando regularmente sua política contra uma métrica de sucesso previamente definida, a fim de verificar se são necessários ajustes na modelagem do ambiente, no sistema de recompensa, nos parâmetros do algoritmo ou na implementação do agente.
- **Teste**: uma vez que o treinamento tenha sido concluído, o modelo será testado para avaliar o quão bem ele aprendeu a tarefa. Isso pode ser feito colocando o agente em um ambiente novo ou mais difícil e medindo a qualidade de seu desempenho.

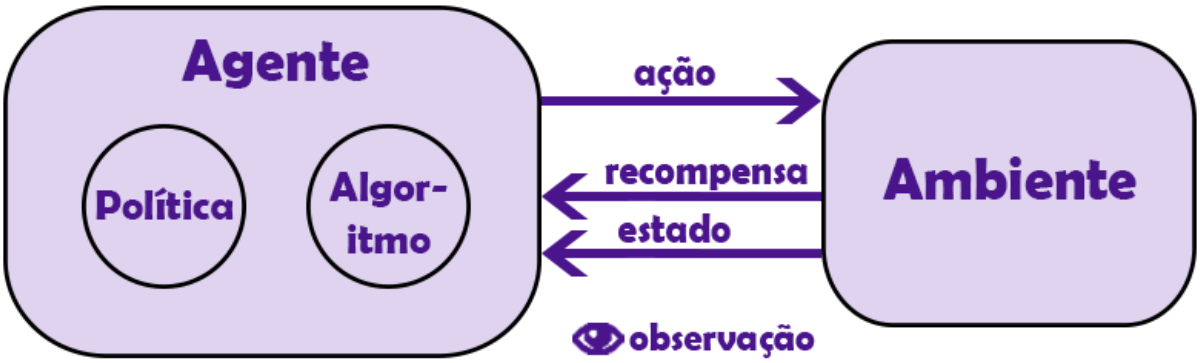

Figura 2.14: Fluxograma exemplificado de Aprendizado por Reforço

Vantagens do Aprendizado por Reforço:

- Consegue aprender por "tentativa e erro" através da interação com um ambiente, sem a necessidade de um conjunto de dados previamente rotulado;
- Pode ser usado para resolver problemas muito complexos, que não podem ser resolvidos por técnicas convencionais;
- O próprio modelo pode se autocorrigir em relação a erros ocorridos durante seu treinamento;
- Consegue obter dados através de interação com o próprio ambiente (que pode ser um ambiente simulado);
- Aprende em tempo real.

Desvantagens do Aprendizado por Reforço:

- Definir as políticas e recompensas pode ser muito complexo;
- Não é ideal para problemas simples ou de domínio muito específico;
- É um processo que demanda uma entrada constante de dados e um poder computacional muito alto.

Aplicações e casos de uso reais:

O *Aprendizado por Reforço* é muito usado para gerar IAs de jogos de computador e videogames. Você pode fazer o computador aprender a jogar praticamente qualquer tipo de jogo, tanto de videogames mais antigos, como o Atari[50] , quanto inteligências mais recentes[51] . Lembra da história do *AlphaGo*, sistema do *Google* que chocou o mundo ao vencer Lee Sedol, o campeão mundial do jogo Go? Ele foi treinado usando essa técnica (combinada com outros métodos).[52]

Ela também é usada extensivamente na área da robótica, por exemplo, em robôs que trabalham em armazéns, como os da *Amazon*[53] e da *Fanuc*[54] . Isso é algo que causou uma grande mudança de paradigma nessas empresas (e, consequentemente, uma possível diminuição da quantidade de trabalhadores humanos necessários nesses locais).

Veículos autônomos[55] e drones[56] também utilizam modelos de *Aprendizado por Reforço* para aprenderem a se movimentar em ambientes como ruas, estradas e pelo ar, analisando o ambiente ao seu redor.

Uma variação poderosa dessa técnica, que se tornou fundamental para o treinamento de modelos de linguagem modernos, é o "*Aprendizado por Reforço a partir de Feedback Humano (RLHF)*", no qual a "recompensa" vem da aprovação de uma pessoa física ou de um algoritmo computacional. Veremos isso em mais detalhe nos próximos capítulos.

Essa ideia de aprender por pura tentativa e erro, sem um "manual de instruções" predefinido, toca em um ponto central da filosofia da IA moderna, brilhantemente resumido por um dos pais do *Aprendizado por Reforço* e ganhador do prêmio Turing (considerado o Nobel da computação), Richard Sutton, em um ensaio chamado "*A Lição Amarga*" (*The Bitter Lesson*)[57] .

A lição, que ele diz que nós pesquisadores demoramos décadas para aprender, é um tanto quanto humilhante: historicamente, os maiores avanços em IA não vieram de nós tentando embutir nosso conhecimento complexo e "inteligente" nas máquinas. Pelo contrário, eles vieram de métodos muito mais simples e genéricos, como o *Aprendizado por Reforço*, que são capazes de escalar massivamente com o poder computacional. Em outras palavras, em vez de darmos ao Eustáquio um mapa do labirinto que nós desenhamos, a abordagem mais poderosa é simplesmente dar a ele um labirinto gigante e poder computacional suficiente para que ele explore cada canto milhões de vezes. A "amarga" verdade é que a solução que ele encontrará sozinho, através da força bruta da exploração, será quase sempre superior à nossa estratégia humana "inteligente". Essa lição é a razão pela qual a IA hoje se baseia em modelos gigantes e em um volume colossal de dados: a escala do aprendizado supera a inteligência artesanal humana.

2.4 Redes Neurais e Deep Learning

Você se lembra que eu havia prometido anteriormente explicar melhor esse tema? Pois bem, vamos começar. *Redes Neurais Artificiais* são um tipo de modelo de *Aprendizado de Máquina* que consiste em simular a estrutura e o comportamento do cérebro humano para realizar o aprendizado.

E você pode me perguntar: "*Mas como assim, simular o cérebro humano?*". Ao menos, eu me fiz essa pergunta quando ouvi essa

explicação pela primeira vez (e claro, essa analogia biológica é só uma inspiração, as redes neurais artificiais são muito mais simples que o cérebro humano, mas vamos lá). Lembrando das aulas de Biologia da escola, a informação trafega no nosso cérebro entre neurônios através de sinapses, certo? Nosso cérebro possui cerca de 80 a 100 bilhões de neurônios com trilhões de conexões.

Uma "Rede Neural Artificial" (que chamarei de agora em diante apenas de **Rede Neural**) tenta simular essa estrutura, contendo uma série de unidades de processamento, chamadas de **neurônios**, organizadas em **camadas** sequenciais. E aí, cada uma dessas conexões, tais quais as sinapses do nosso cérebro, pode transmitir um sinal (informação, que aqui serão dados numéricos) aos neurônios da camada subsequente (ou, às vezes, de camadas anteriores ou até da mesma, mas isso são detalhes).

Essas redes são geralmente compostas de:

- Uma camada de entrada, que recebe os dados brutos (por exemplo, os pixels de uma imagem);
- Zero ou mais camadas intermediárias "ocultas", que fazem os cálculos;
- Uma camada de saída, que fornecerá o resultado do modelo (por exemplo, a classificação da imagem).

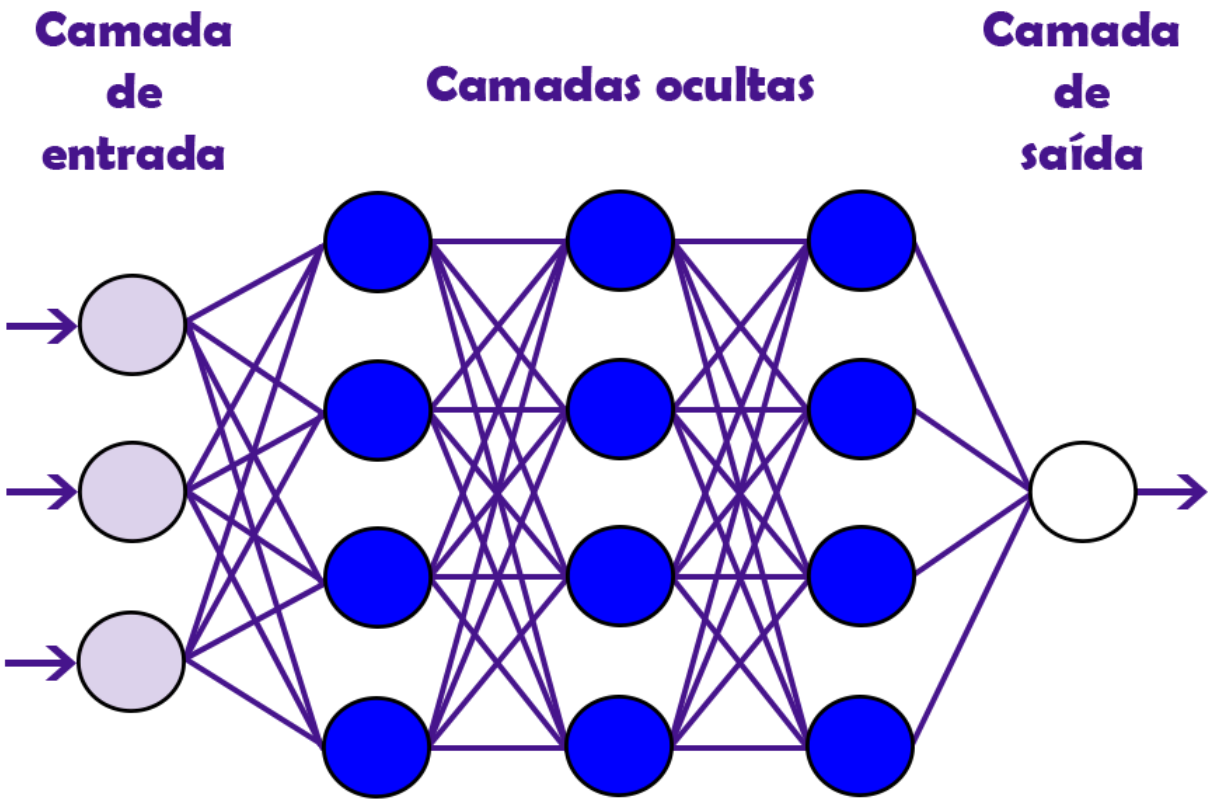

Figura 2.15: Exemplo de Rede Neural

Sei que pode parecer confuso, mas os neurônios nada mais fazem do que uma operação matemática bem simples: em termos simples, aquele "*y = ax + b*" que aprendemos na escola. Cada neurônio é responsável por receber uma entrada, processá-la (ou seja, fazer as operações matemáticas) e transmitir o seu resultado adiante. Neurônios que estejam na camada de entrada receberão dados brutos externos, como imagens ou documentos de texto, convertidos para valores numéricos. Já neurônios de camadas intermediárias receberão o resultado do processamento dos neurônios de uma camada anterior.

Cada conexão entre neurônios tem um **peso** (um valor numérico correspondente à entrada) associado, que é ajustado durante o processo de aprendizado. Os pesos de suas redes neurais são basicamente a base de conhecimento das modelos de IA — alterá-los alterará também as suas capacidades.

Neurônios também possuem um **viés** (bias), que funciona como um valor de ajuste adicional, e duas funções principais:

- **Função de soma**: calcula a soma ponderada dos valores de entrada multiplicados por seus respectivos pesos e somando esse resultado ao viés;
- **Função de ativação**: é uma função matemática, quase sempre não linear, aplicada ao valor que chega em um neurônio, após a soma ponderada das entradas. Ela transforma esse valor de forma que a rede possa aprender padrões mais complexos, e não apenas combinações lineares simples. Diferentes funções produzem diferentes tipos de transformação. Por exemplo, se aquele neurônio estiver utilizando a popular função de ativação *ReLU*, qualquer valor negativo vira zero, e valores positivos passam adiante sem alteração. Já se ele estiver com a função *Sigmoide*, o valor é "espremido" para ficar entre 0 e 1, representando um grau de ativação e, em certos casos, pode até ser interpretado como uma probabilidade.

Esse processo é essencial para permitir que a rede aprenda representações complexas dos dados de entrada. De maneira simplificada, o que cada neurônio faz é aplicar uma equação matemática às entradas que recebe, processando esse valor e enviando o resultado adiante. E quando muitos neurônios fazem isso em conjunto, organizados em várias camadas, a rede se torna capaz de resolver problemas bastante complexos.

Mas como a rede aprende a fazer isso bem?

Durante o treinamento, a rede ajusta os pesos e vieses de suas conexões para melhorar suas previsões. Esse processo é baseado em uma medida chamada **função de perda**, que calcula o quão distante a saída produzida pela rede está da saída esperada. Em tarefas supervisionadas, por exemplo, essa saída esperada vem dos rótulos dos dados de treino. Já em tarefas não supervisionadas ou autossupervisionadas, essa "resposta correta" pode ser gerada pelo

próprio modelo, como no caso de reconstruir uma imagem original a partir de uma versão embaralhada (*Autoencoder*) ou prever uma palavra que foi ocultada (como no *BERT*).

E aí, para saber como alterar os pesos e vieses de maneira eficiente, a rede usa um algoritmo chamado "**backpropagation**"[58] (podemos traduzir como "retropropagação do erro"), que é uma das chaves que viabilizou o Deep Learning moderno. O *backpropagation* é o algoritmo que vai tentar "encontrar os culpados" por um erro.

Vamos imaginar que os dados entraram na primeira camada da rede. Eles vão sendo processados ao longo dela até chegarem à camada de saída, que fará uma previsão (por exemplo, que se tratava da imagem de um gato, não de um cachorro ou uma girafa) e, nesse ponto, o algoritmo calculará o erro total na camada de saída. Em seguida, ele começa a propagar esse valor de erro para trás, camada por camada da rede neural, calculando o quanto cada peso e viés contribuiu para o erro final da rede. Isso é feito por meio de uma técnica de cálculo matemático chamada "regra da cadeia" e, dessa forma, ele consegue determinar uma informação chamada de **gradiente** da função de perda.

Os gradientes são, de forma simplificada, a taxa de variação da função de perda em relação a cada peso e viés da rede, ou seja, basicamente o quanto cada peso e viés contribuiu para esse erro final, e indicam em que direção e com que intensidade cada parâmetro (os pesos e vieses) da rede deve ser ajustado para reduzir o erro. Esses gradientes são então usados por um otimizador (como o *Gradiente Descendente*, *Adam*, *AdamW*, *RMSProp*, *Muon*, entre outros) para ajustar os pesos e minimizar o erro da rede ao longo do tempo.

Se o gradiente de um peso da rede for um número positivo e alto, por exemplo, isso quer dizer que diminuir esse peso deve reduzir o erro (e aí, o otimizador saberá quanto deve ajustá-lo). É como seguir uma trilha de montanha tentando descer o mais rápido

possível: o gradiente diz para que lado o terreno desce mais íngreme, e o otimizador dá o passo.

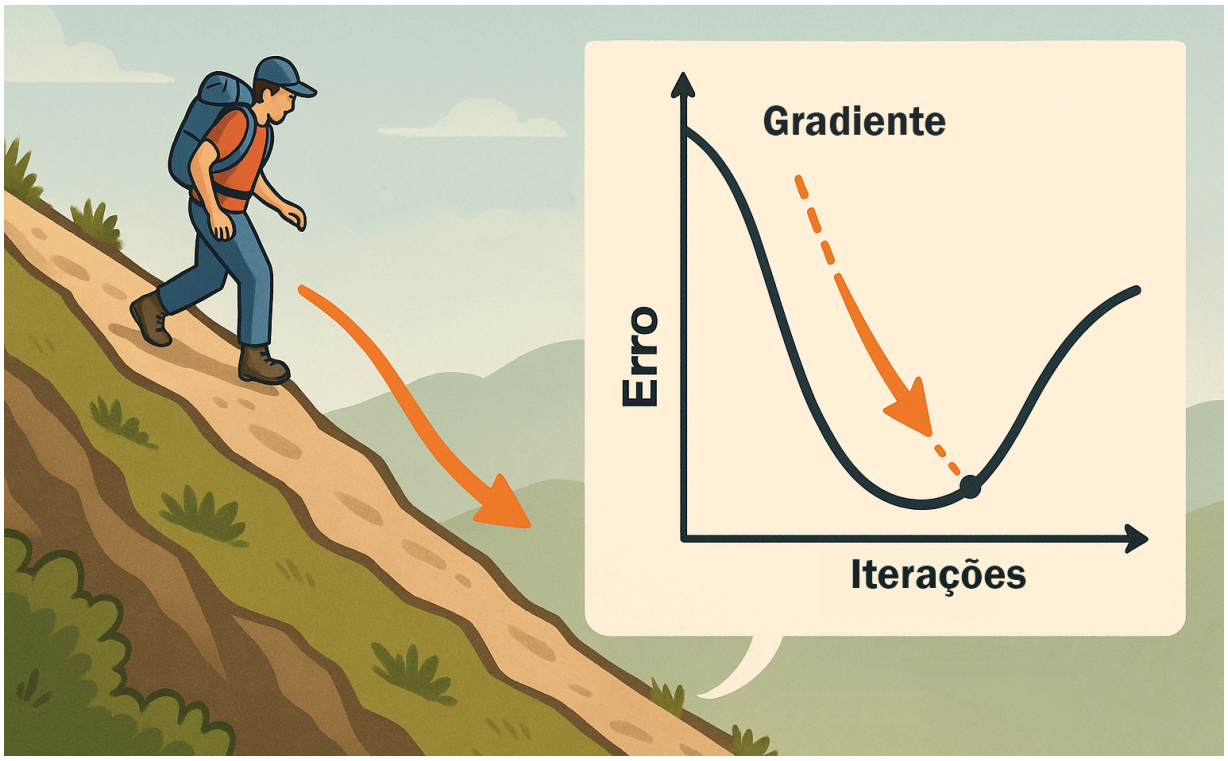

Figura 2.16: Imagem ilustrando uma rede neural minimizando o erro

O processo todo — calcular os gradientes com o backpropagation e depois atualizá-los com o otimizador — é repetido milhares (ou até milhões) de vezes durante o treinamento, até que a rede consiga fazer boas previsões.

O **Deep Learning** ou **Aprendizado Profundo** é assim chamado pois se baseia em redes neurais com múltiplas camadas intermediárias ocultas (geralmente duas ou mais), dando profundidade à rede. Ele nada mais é do que um nome mais elegante para redes neurais com muitas camadas intermediárias. Essa profundidade permite que a rede aprenda representações mais abstratas e hierárquicas dos dados, o que é especialmente útil em tarefas complexas como reconhecimento de imagens, tradução automática ou geração de texto. Além da profundidade, o termo Deep Learning também carrega a ideia de treinar essas redes em

larga escala, com grandes volumes de dados e alto poder computacional, usando arquiteturas modernas como CNNs e Transformers.

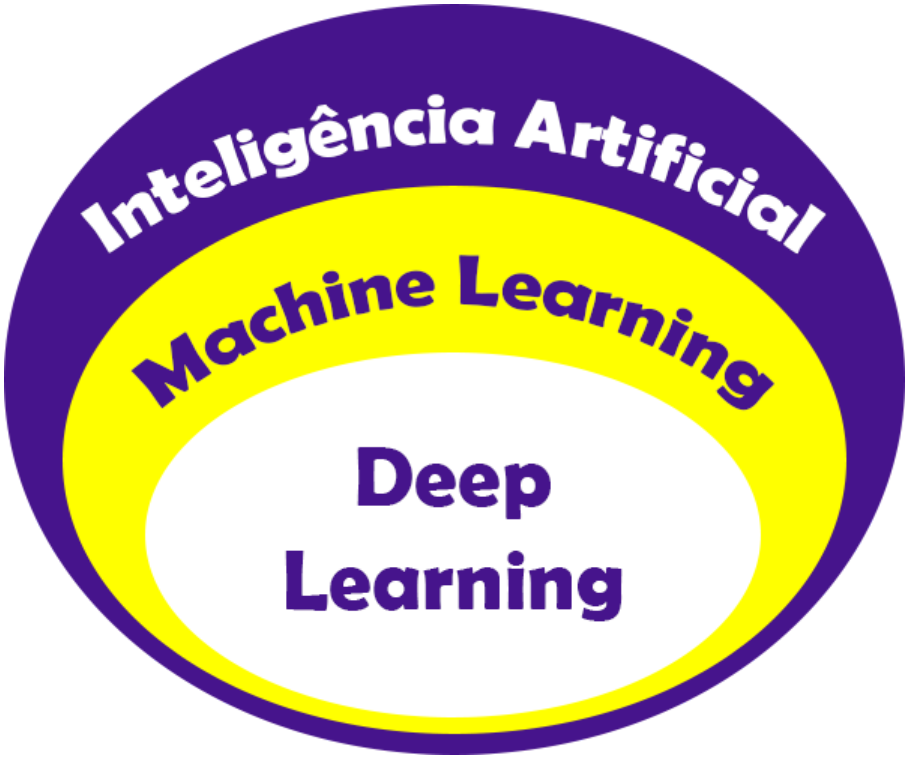

Figura 2.17: Comparação de Inteligência Artificial, Machine Learning e Deep Learning

Tá, mas e na prática? Quando falamos de reconhecimento de imagens, por exemplo, suponha que tenhamos várias imagens de três formas geométricas — triângulos, círculos e quadrados — e queiramos treinar uma rede neural para que, dada uma imagem de uma forma geométrica, ela identifique qual é essa forma. Todas as imagens estão em preto-e-branco e possuem 30 pixels de altura e de largura.

Nesse caso, poderíamos ter 900 neurônios na camada de entrada, um para cada pixel da imagem (30 vezes 30), e cada neurônio de entrada adotaria um valor de 0 ou 1 — 0 para pixels pretos (o fundo da imagem) e 1 para pixels brancos (os que contêm alguma parte da imagem). Já na camada de saída, poderíamos ter 3 neurônios, um para cada tipo de forma geométrica que queremos classificar. São as

camadas ocultas intermediárias que descobrirão as características dessa imagem, como "bordas" das formas, contornos etc.

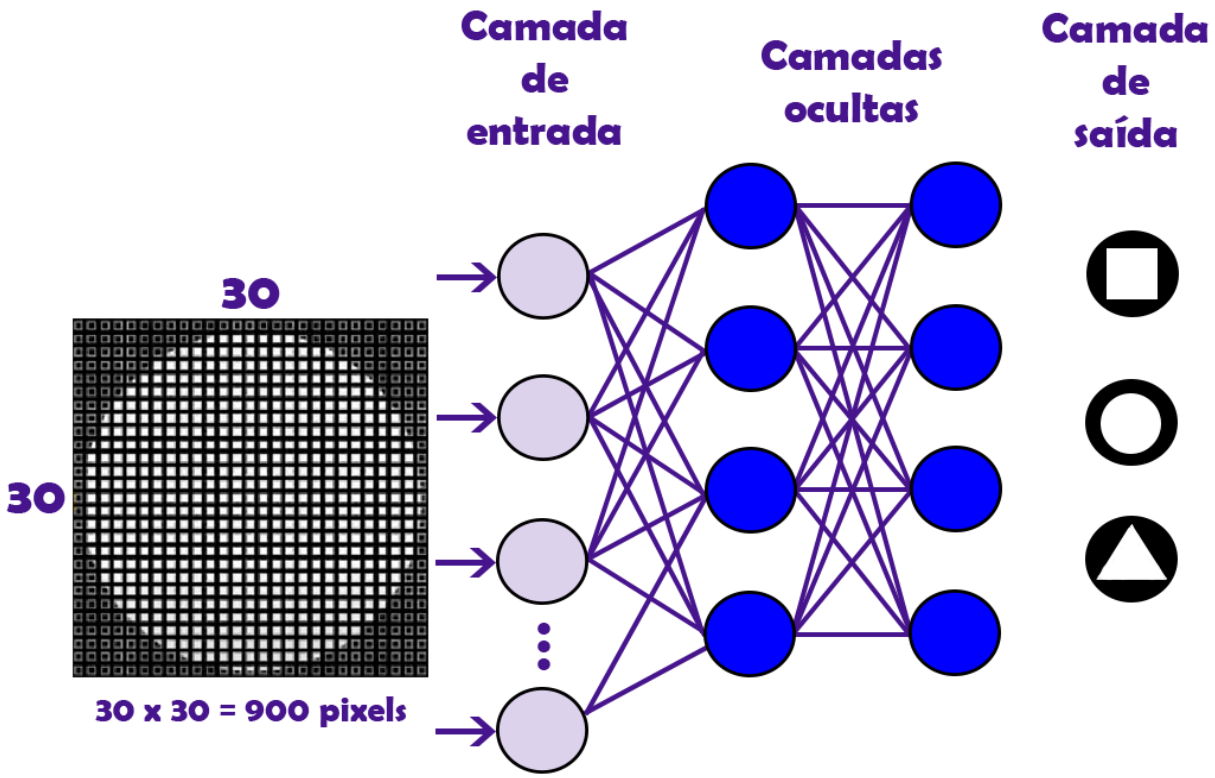

Figura 2.18: Exemplo de uma rede neural para classificação de formas geométricas

Tipos de Redes Neurais

Existem diversas arquiteturas de redes neurais, cada uma projetada para lidar melhor com certo tipo de tarefa. A seguir, vou citar alguns dos principais tipos:

1. **Redes Neurais Feedforward (FNN — *Feedforward Neural Network*)**[59] : esse é o tipo mais básico de rede neural. As informações são transmitidas em uma única direção, no sentido da entrada para a saída. Não há *loops* no caminho, daí o nome "*feedforward*", que podemos traduzir livremente como "alimentação avante" ou "alimentação para a frente". Também podem ser chamadas de "perceptrons multicamadas (MLP)", um nome encontrado bem frequentemente.

2. **Redes Neurais Convolucionais (CNN — *Convolutional Neural Network*)**[60] : é um tipo de *FNN* usado principalmente para processar imagens. Elas são chamadas de "convolucionais" por utilizarem uma operação matemática chamada "convolução", muito usada em matrizes e excelente para detectar padrões espaciais em imagens, como bordas e contornos. São muito usadas em problemas de Visão Computacional, e foram usadas na *AlexNet*, rede neural que venceu um concurso de visão computacional em 2012, iniciando a era do Deep Learning em visão.

3. **Redes Neurais Recorrentes (RNN — *Recurrent Neural Network*)**[61] : diferentemente das *FNNs*, onde o resultado de um neurônio é transmitido apenas para neurônios de camadas subsequentes, as *RNNs* permitem *loops*, ou seja, a saída de um nó pode voltar para a entrada do próprio nó. Com isso, ela consegue levar em consideração o que viu anteriormente, ocasionando um tipo de "memória". Essa propriedade as torna adequadas para tarefas que envolvem sequências, como tradução automática ou geração de texto.

4. **Redes de Memória Longa de Curto Prazo (LSTM — *Long Short-Term Memory*)**[62] : é um tipo especial de *RNN* que pode aprender dependências de longo prazo. As redes *LSTM* têm a capacidade de remover ou adicionar informações ao estado da célula através de estruturas chamadas "portas" (*gates*), que definem quais informações serão lembradas, descartadas ou passadas adiante. Elas são usadas em tarefas que exigem lembrar de informações por um longo período de tempo, como previsão de séries temporais, tradução automática, reconhecimento de voz, entre outros.

5. **AutoEncoders**[63] : são redes neurais compostas de duas partes principais — o *Encoder* e o *Decoder*. O *Encoder* recebe os dados de entrada, aprende suas características mais importantes e os comprime em uma versão de dimensão reduzida. O *Decoder*

receber a representação codificada e tenta reconstruir os dados de entrada originais a partir dela. Ou seja, ele tenta reverter o processo de codificação. O objetivo é treinar o *AutoEncoder* de modo que a saída (os dados reconstruídos) seja o mais próximo possível dos dados de entrada originais e, com isso, possa ser generalizado para outros dados.

6. **Redes Adversárias Generativas (GAN — *Generative Adversarial Network*)**[64] : uma rede *GAN* pode ser pensada como duas redes neurais concorrentes — uma **rede geradora**, que aprende a gerar novos dados semelhantes aos de treinamento; e uma **rede discriminadora**, que tenta distinguir os dados gerados dos reais. O processo de treinamento das redes e de *feedback* é realizado repetidas vezes, com o objetivo de que a rede geradora se torne cada vez melhor em produzir dados falsos que pareçam reais, e de que a rede discriminadora, por sua vez, se torne cada vez melhor em identificar esses dados falsos. *GANs* são aplicadas em uma variedade de aplicações, incluindo aprimoramento de imagens, geração de imagens a partir de ruído ou texto, melhora da resolução etc.

7. **Transformers**[65] : lembre-se deste nome. Eles são uns dos grandes responsáveis pela revolução dos *LLMs* (*Large Language Models*), como o *ChatGPT*, que estamos vivendo atualmente. Os *Transformers* são um tipo de arquitetura de rede neural introduzido no artigo "*Attention is All You Need*"[66] do *Google*. O principal conceito por trás da arquitetura de um *Transformer* é o mecanismo de "atenção", que permite que o modelo "preste mais atenção" a algumas palavras do que a outras, o que resulta em uma maior capacidade de entender o contexto e a sintaxe de um texto.

   Ao contrário das *RNNs*, que processam sequências de dados de forma sequencial, uma rede baseada em *Transformers* pode processar todos os pontos de dados de entrada de forma paralela, o que a torna muito mais eficiente. Além disso, essa abordagem paralela ajuda a lidar com o "problema de longa

dependência", comum nas RNNs, onde a rede tem dificuldade em aprender a conectar informações relevantes em sequências longas. Isso é especialmente útil para tarefas de Processamento de Linguagem Natural de tradução ou geração de texto, que necessitam de uma compreensão mais precisa do contexto de frases e parágrafos anteriores.

Bastante coisa, não é? Este é, realmente, um capítulo mais denso em termos de informação, onde foram apresentados os nomes de diversos conceitos e tecnologias, que representam apenas uma parte desse mundo de Inteligência Artificial. Ele é a base do que vamos discutir nos próximos capítulos.

Vimos que o jeito que o Scooby aprende, que a Paulinha generaliza e que o Pedrinho descobre padrões têm paralelos diretos com os diferentes tipos de aprendizado. Esses princípios, quando escalados com redes neurais profundas, dão origem aos sistemas impressionantes de hoje em dia.

Vários desses conceitos serão mencionados novamente e, conhecendo-os agora, você poderá compreender mais facilmente como eles são aplicados em sistemas de IA generativa e grandes modelos de linguagem, os famosos *LLMs*, que serão o foco do próximo capítulo.

Referências bibliográficas:

[1] ALPAYDIN, Ethen. *Introduction to Machine Learning.* 4. ed. Cambridge (EUA): The MIT Press, 2020.

[2] DAMACENO, Laura. Regressão Linear?. *Medium*, 2020. Disponível em:

https://medium.com/@lauradamaceno/regress%C3%A3o-linear-6a7f247c3e29

[3] MATSUMOTO, Fernando. Modelos de Predição | Regressão Logística. *Medium*, 2019. Disponível em: https://medium.com/turing-talks/turing-talks-14-modelo-de-predi%C3%A7%C3%A3o-regress%C3%A3o-log%C3%ADstica-7b70a9098e43

[4] CAMPOS, Raphael. Árvores de Decisão. *Medium*, 2017. Disponível em: https://medium.com/machine-learning-beyond-deep-learning/%C3%A1rvores-de-decis%C3%A3o-3f52f6420b69

[5] DONGES, Niklas; SILVA, Josenildo C. Aprendendo em uma Floresta Aleatória. *Medium*, 2017. Disponível em: https://medium.com/machina-sapiens/o-algoritmo-da-floresta-aleat%C3%B3ria-3545f6babdf8

[6] LAGO, Maria E. F. Aprendizado de Máquina: Máquina de Vetores de Suporte ('Support Vector Machine' — SVM). *Medium*, 2021. Disponível em: https://mariaef-lago.medium.com/algoritmo-de-m%C3%A1quina-de-vetores-de-suporte-support-vector-machine-svm-10773e50a06

[7] CRUVINEL, Leandro. Ciência de dados com k-NN na

prática. *Medium*, 2018. Disponível em: https://leandrocruvinel.medium.com/ci%C3%AAncia-de-dados-com-k-nn-na-pr%C3%A1tica-f78bc516a67a

[8] TAMAIS, Ana L. M. Modelos de Predição | Naive Bayes. *Medium*, 2019. Disponível em: https://medium.com/turing-talks/turing-talks-16-modelo-de-predi%C3%A7%C3%A3o-naive-bayes-6a3e744e7986

[9] CARVALHO, J. C. N. Sobre o algoritmo Xgboost. *Medium*, 2020. Disponível em: https://joaoclaudionc.medium.com/obre-o-algoritmo-xgboost-139305d1f1aa

[10] AZAMBUJA, Pedro. AdaBoost (Adaptive Boosting). *Medium*, 2020. Disponível em: https://pedroazambuja.medium.com/adaboost-adaptive-boosting-dbbec150fced

[11] TELES, Thiago. Explorando loss function em Aprendizado de Máquina para problemas de regressão. *LinkedIn*, 2023. Disponível em: https://www.linkedin.com/pulse/explorando-loss-function-em-aprendizado-de-m%C3%A1quina-para-thiago-teles/

[12] IBM. *O que é sobreajuste?*. c2023. Disponível em: https://www.ibm.com/br-pt/topics/overfitting

[13] KHATRI, Samidha; ARORA, Aishwarya; AGRAWAL, Arun P. Supervised Machine Learning Algorithms for Credit Card Fraud Detection: A Comparison. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2020, pp. 680-683. 2020. Disponível em: https://ieeexplore.ieee.org/abstract/document/9057851

[14] DUA, Prerna; BAIS, Sonali. Supervised Learning Methods for Fraud Detection in Healthcare Insurance. In: DUA, S., ACHARYA, U., Dua, P. (eds.). Machine Learning in Healthcare Informatics. *Intelligent Systems Reference Library*, v. 56, p. 261-285. Springer, Berlin, Heidelberg, 2014. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-40017-9_12

[15] SILVA, Bárbara; LAURETTO, Marcelo; ANDRADE, Pablo; ARAÚJO, Luciana. Avaliação de uma Abordagem de Aprendizado Supervisionado para Operações no Mercado de Ações. In: Simpósio Brasileiro de Sistemas de Informação (SBSI), 9. ,

2013, João Pessoa. *Anais* [...]. Porto Alegre: Sociedade Brasileira de Computação, 2013. p. 109-120. Disponível em: https://sol.sbc.org.br/index.php/sbsi/article/view/5680/5577

[16] CHOI, Lennon H. T.; HO, Winky K. O. The Use of Machine Learning in Real Estate Research. *Land*, v. 12, n. 4, 2023. Disponível em: https://www.mdpi.com/2073-445X/12/4/740

[17] GOMEZ-URIBE, Carlos A.; HUNT, Neil. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems*, v. 6, n. 4, 2015. Disponível em: https://dl.acm.org/doi/10.1145/2843948

[18] SANTANA, Roniel V.; PONTES, Heráclito L. J. Aplicação da Clusterização por K-means para Criação de Sistema de Recomendação de Produtos baseado em Perfis de Compra. *Navus: Revista de Gestão e Tecnologia*, n. 10, 2020. Disponível em: https://dialnet.unirioja.es/servlet/articulo?codigo=7774814

[19] CHATURVEDI, Akshay K.; PELEJA, Filipa; FREIRE, Ana. Recommender System for News Articles using

Supervised Learning. *arXiv*, 2017. Disponível em: https://arxiv.org/abs/1707.00506

[20] GOOGLE DEVELOPERS. *O que é clustering?*. c2022. Disponível em: https://developers.google.com/machine-learning/clustering/overview?hl=pt-br

[21] ANASTACIO, Bruno. K-means: o que é, como funciona, aplicações e exemplo em Python. *Medium*, 2020. Disponível em: https://medium.com/programadores-ajudando-programadores/k-means-o-que-%C3%A9-como-funciona-aplica%C3%A7%C3%B5es-e-exemplo-em-python-6021df6e2572

[22] PATLOLLA, Chaitanya R. Understanding the concept of Hierarchical clustering Technique. *Medium*, 2018. Disponível em: https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec

[23] MONTEIRO, Gabriel. Entendendo DBSCAN. *Medium*, 2020. Disponível em: https://gabriellm.medium.com/entendendo-dbscan-770f680d9160

[24] HENRIQUE, Alexandre. O raciocínio por trás do Algoritmo Expectation-Maximization. *Medium*, 2021. Disponível em: https://medium.com/b2w-engineering/o-racioc%C3%ADonio-por-tr%C3%A1s-do-algoritmo-expectation-maximization-91d4a8588778

[25] VERAS, Felipe. Abrindo a Caixa Preta: PCA (Análise de Componentes Principais). *Medium*, 2021. Disponível em: https://medium.com/@felipeverasaraujo/abrindo-a-caixa-preta-pca-an%C3%A1lise-de-componentes-principais-d5d400781dfe

[26] MILLS, Peter. Singular Value Decomposition (SVD) Tutorial: Applications, Examples, Exercises. *Medium*, 2017. Disponível em: https://medium.com/cube-dev/singular-value-decomposition-tutorial-52c695315254

[27] DUKARE, Agasti K. Anomaly Detection in Python with Gaussian Mixture Models. *Medium*, 2020. Disponível em: https://towardsdatascience.com/understanding-anomaly-detection-in-python-using-gaussian-mixture-model-e26e5d06094b

[28] TEODORO, Rafael. Clusterização utilizando Mean Shift como método para a criação de rótulos em imagens. *Medium*, 2023. Disponível em: https://medium.com/@rafael.spteo/clusteriza%C3%A7%C3%A3o-utilizando-mean-shift-como-m%C3%A9todo-para-a-cria%C3%A7%C3%A3o-de-r%C3%B3tulos-em-imagens-e84e6b2a8ce6

[29] COSTA, Bernardo. Uma Introdução ao Algoritmo Apriori. *Medium*, 2019. Disponível em: https://medium.com/@bernardo.costa/uma-introdu%C3%A7%C3%A3o-ao-algoritmo-apriori-60b11293aa5a

[30] SESTINI, Paulo. Redes Neurais | Autoencoders com PyTorch. *Medium*, 2020. Disponível em: https://medium.com/turing-talks/redes-neurais-autoencoders-com-pytorch-fbce7338e5de

[31] IBM. Aplicações de aprendizado não supervisionado. In: *O que é Aprendizado não Supervisionado?*. c2023. Disponível em: https://www.ibm.com/br-pt/topics/unsupervised-learning

[32] KRIKORIAN, Mauro. Fraud Detection applying

Unsupervised Learning techniques. *Medium*, 2021. Disponível em: https://medium.com/southworks/fraud-detection-applying-unsupervised-learning-techniques-4ae6f71b266f

[33] CARCILLO, Fabrizio; LE BORGNE, Yann-Aël; CAELEN, Olivier; KESSACI, Yacine; OBLÉ, Fréderic; BONTEMPI, Gianluca Combining unsupervised and supervised learning in credit card fraud detection. *Information Sciences*, v. 557, p. 317-331, 2021. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0020025519304451

[34] ALIBABA CLOUD. *Applying Unsupervised Machine Learning Techniques to Build End-to-End Customer Segmentation Solution*. 2021. Disponível em: https://www.alibabacloud.com/blog/applying-unsupervised-machine-learning-techniques-to-build-end-to-end-customer-segmentation-solution_597462

[35] FERRUZ, Noelia; SCHMIDT, Steffen; HÖCKER, Birte. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, n. 13, art. 4348, 2022. Disponível em: https://www.nature.com/articles/s41467-022-32007-

[36] AMATRIAIN, Xavier; BASILICO, Justin. Netflix Recommendations: Beyond the 5 stars (Part 2). *Netflix TechBlog*, 2012. Disponível em: https://netflixtechblog.com/netflix-recommendations-beyond-the-5-stars-part-2-d9b96aa399f5

[37] JOTHI PRAKASH, V.; NITHYA, L. M. A Survey On Semi-Supervised Learning Techniques. *International Journal of Computer Trends and Technology*, v. 8, n. 1, 2014. Disponível em: https://arxiv.org/ftp/arxiv/papers/1402/1402.4645.pdf

[38] DOBILAS, Saul. Semi-Supervised Learning — How to Assign Labels with Label Propagation Algorithm. *Towards Data Science*, 2021. Disponível em: https://towardsdatascience.com/semi-supervised-learning-how-to-assign-labels-with-label-propagation-algorithm-9f1683f4d0eb

[39] STEEN, Doug. A Gentle Introduction to Self-Training and Semi-Supervised Learning. *Towards Data Science*, 2020. Disponível em: https://towardsdatascience.com/a-gentle-introduction-to-self-training-and-semi-supervised-

learning-ceee73178b38

[40] DU, Jun; LING, Charles S.; ZHOU, Zhi-Hua. When Does Co-Training Work in Real Data? *IEEE Transactions on Knowledge and Data Engineering*, v. 23, n. 5, pp. 788-799, 2011. Disponível em: https://ieeexplore.ieee.org/document/5560662

[41] SETTOUTI, Nesma; EL HABIB DANO, Mostafa; EL AMINE LAZOUNI, Mohammed; CHIKH, Mohammed A. Random forest in semi-supervised learning (Co-Forest). *8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*, Algiers, Algeria, p. 326-329, 2013. Disponível em: https://ieeexplore.ieee.org/abstract/document/6602385

[42] CHOUDHRY, Amit. Facebook's DeepFace Software Can Match Faces With 97.25% Accuracy. *Forbes*, 2014. Disponível em: https://www.forbes.com/sites/amitchowdhry/2014/03/18/facebooks-deepface-software-can-match-faces-with-97-25-accuracy/

[43] CALANCA, Paulo; MATHEUS, Yuri; RAPHAELL, Bruno. Quais são os 4 tipos de aprendizagem na IA, algoritmos e usos no dia a dia. *Alura*, 2023.

Disponível em: https://www.alura.com.br/artigos/quais-sao-tipos-aprendizagem-ia-inteligencia-artificial

[44] SALIAN, Isha. SuperVize Me: What's the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? *NVIDIA Blog*, 2018. Disponível em: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

[45] ZHANG, Bowen; CAO, Songjun; ZHANG, Xiaoming; ZHANG, Yike; MA, Long; SHINOZAKI, Takahiro. Censer: Curriculum Semi-supervised Learning for Speech Recognition Based on Self-supervised Pre-training. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2206.08189

[46] FUKUSHIMA, William; REIS, Lucas O. Aprendizado por Reforço #2 | Processo de Decisão de Markov — Parte 1. *Medium*, 2020. Disponível em: https://medium.com/turing-talks/aprendizado-por-refor%C3%A7o-2-processo-de-decis%C3%A3o-de-markov-mdp-parte-1-84e69e05f007

[47] ADL. An introduction to Q-Learning: reinforcement learning. *freeCodeCamp.org*, 2018. Disponível em:

https://www.freecodecamp.org/news/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc/

[48] KAPOOR, Sanyam. Policy Gradients in a Nutshell. *Towards Data Science*, 2018. Disponível em: https://towardsdatascience.com/policy-gradients-in-a-nutshell-8b72f9743c5d

[49] DOBILAS, Saul. Reinforcement Learning with SARSA — A Good Alternative to Q-Learning Algorithm. *Towards Data Science*, 2022. Disponível em: https://towardsdatascience.com/reinforcement-learning-with-sarsa-a-good-alternative-to-q-learning-algorithm-bf35b209e1c

[50] KAISER, Lukasz; BABAEIZADEH, Mohammad; MILOS, Piotr; OSINSKI, Blazej; CAMPBELL, Roy H.; CZECHOWSKI, Konrad; ERHAN, Dumitru; FINN, Chelsea; KOZAKOWSKI, Piotr; LEVINE, Sergey; MOHIUDDIN, Afroz; SEPASSI, Ryan; TUCKER, George; MICHALEWSKI, Henryk. Model-Based Reinforcement Learning for Atari. *arXiv*, 2019. Disponível em: https://arxiv.org/abs/1903.00374

[51] LAMPLE, Guillaume; CHAPLOT, Devendra S. Playing FPS Games with Deep Reinforcement

Learning. *Thirty-First AAAI Conference on Artificial Intelligence*, v. 31, n. 1, 2017. Disponível em: https://ojs.aaai.org/index.php/AAAI/article/view/10827

[52] SILVER, David; SCHRITTWIESER, Julian; SIMONYAN, Karen; ANTONOGLOU, Ioannis; HUANG, Aja; GUEZ, Arthur; HUBERT, Thomas; BAKER, Lucas; LAI, Matthew; BOLTON, Adrian; CHEN, Yutian; LILLICRAP, Timothy; HUI, Fan; SIFRE, Laurent; VAN DEN DRIESSCHE, George; GRAEPEL, Thore; HASSABIS, Demis. Mastering the game of Go without human knowledge. *Nature*, v. 550, n. 7676, p. 354–359, 2017. Disponível em: https://www.nature.com/articles/nature24270

[53] SCHMELZER, Ron. Amazon Dives Deep into Reinforcement Learning. *Forbes*, 2019. Disponível em: https://www.forbes.com/sites/cognitiveworld/2019/06/14/amazon-dives-deep-into-reinforcement-learning/?sh=37bd13455fe0

[54] KNIGHT, Will. This Factory Robot Learns a New Job Overnight. *MIT Technology Review*, 2016. Disponível em: https://www.technologyreview.com/2016/03/18/161

519/this-factory-robot-learns-a-new-job-overnight/

[55] KIRAN, B. R.; SOBH, Ibrahim; TALPAERT, Victor; MANNION, Patrick; AL SALLAB, Ahmed; YOGAMANI, Senthil; PÉREZ, Patrick. Deep Reinforcement Learning for Autonomous Driving: A Survey. *arXiv*, 2021. Disponível em: https://arxiv.org/abs/2002.00444

[56] MILLÁN-ARIAS, Cristian; CONTRERAS, Ruben; CRUZ, Francisco; FERNANDES, Bruno. Reinforcement Learning for UAV control with Policy and Reward Shaping. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2212.03828

[57] SUTTON, R. The Bitter Lesson. 2019. Disponível em: http://www.incompleteideas.net/IncIdeas/BitterLesson.html

[58] RUMELHART, David; HINTON, George; WILLIAMS, Ronald. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Disponível em: https://www.nature.com/articles/323533a0

[59] BAGHERI, Reza. An Introduction to Deep

Feedforward Neural Networks. *Towards Data Science*, 2020. Disponível em: https://towardsdatascience.com/an-introduction-to-deep-feedforward-neural-networks-1af281e306cd

[60] MISHRA, Mayank. Convolutional Neural Networks, Explained. *Towards Data Science*, 2020. Disponível em: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[61] NABI, Javaid. Recurrent Neural Networks (RNNs). *Towards Data Science*, 2019. Disponível em: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

[62] HOCHREITER, Sepp; SCHIMIDHUBER, Jürgen. Long Short-term Memory. *Neural Computation*, v. 9, n. 8, 1997. Disponível em: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory

[63] HUBENS, Nathan. Deep inside: Autoencoders. *Towards Data Science*, 2018. Disponível em: https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

[64] ROCCA, Joseph. Understanding Generative

Adversarial Networks (GANs). *Towards Data Science*, 2019. Disponível em: https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

[65] PHI, Michael. Illustrated Guide to Transformers- Step by Step Explanation. *Towards Data Science*, 2020. Disponível em: https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0

[66] VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention Is All You Need. *arXiv*, 2017. Disponível em: https://arxiv.org/abs/1706.03762

Capítulo 3
Modelos de linguagem e como funcionam

Agora que você já entende o que é uma Inteligência Artificial, quais são seus principais tipos e como elas aprendem, podemos finalmente ir mais a fundo nos modelos de linguagem. Neste capítulo, você conhecerá os principais conceitos de *PLN* (Processamento de Linguagem Natural) que levaram ao surgimento dos *LLMs*.

Mas para começar, temos que entender o que, afinal, é um *LLM*. Esse tem sido um dos termos da moda desde o surgimento do *ChatGPT*, e é a sigla de "*Large Language Model*", que podemos traduzir livremente como "Grande Modelo de Linguagem". E para entender o que é um "Grande Modelo de Linguagem" e como eles funcionam, precisamos primeiramente entender o que é um modelo de linguagem.

3.1 O que é um modelo de linguagem?

Vamos começar tirando o elefante da sala. Um **modelo de linguagem** nada mais é do que um modelo probabilístico que, depois de ser treinado com muitos textos, consegue prever a próxima palavra em uma sequência de palavras. Pronto, capítulo terminado, pode ir para o próximo.

Brincadeiras à parte, essa é realmente a base lógica por trás de todos esses modelos sobre os quais ouvimos falar atualmente e seus predecessores. A questão interessante aqui é como eles fazem isso e como esses sistemas conseguiram evoluir ao longo do tempo.

Todo idioma, seja ele português, inglês ou turco, possui padrões e estruturas como verbos e substantivos (ou, se você quiser ser mais

preciso, palavras que representam ações e palavras que representam entidades). Uma criança brasileira aprende que, sempre que ela começar uma frase com "*Eu gosto*", virá na sequência a palavra "*de*", e isso independente de ela saber que a palavra "*de*" é uma preposição (ou mesmo o que raios é uma preposição). Além disso, ela aprende que essa frase está relacionada a um conceito positivo, e que depois do "*de*" virá algo como uma atividade, uma comida ou o nome de uma pessoa.

- Eu gosto de pizza.
- Eu gosto de viajar.

Uma criança inglesa ou estadunidense, no entanto, apesar de aprender igualmente que a frase "*I like*" (tradução de "*Eu gosto*" em inglês) transmite um conceito positivo, nunca verá uma preposição na sequência:

- I like pizza. (literalmente, "*Eu gosto pizza*")
- I like to travel. (literalmente, "*Eu gosto viajar*". Apesar de "to" poder ser uma preposição em alguns casos, neste tipo de frase sua função é apenas de marcador de infinitivo)

Já uma criança turca, além de também absorver que esse é um conceito positivo, aprende naturalmente que verbos (palavras de ação) vão sempre no final da frase. A palavra turca "*severim*" pode ser traduzida como "*(Eu) gosto*", mas diferentemente do português e do inglês, em turco, a "coisa gostada" vem antes do verbo:

- Pizza severim. (literalmente, "*Pizza gosto*")
- Gezmeyi severim. (literalmente, "*Viajar gosto*")

Não, isso não é uma aula de turco (e prometo que não haverá prova no final), mas serve para demonstrar bem que, apesar do fato de que diferentes idiomas podem ter estruturas completamente distintas entre si, todos seguem uma certa estrutura interna e têm padrões, e essas características são aprendidas por seus falantes nativos.

O que um modelo de linguagem tenta fazer é exatamente isso: aprender padrões na linguagem humana, quais palavras geralmente se relacionam com quais outras e qual é a probabilidade de uma determinada palavra ser a próxima da frase, considerando o contexto das anteriores.

3.2 O início e evolução de PLN e modelos de linguagem

Vamos de aula curtinha de história? Não vou contar apenas a história, mas também explicar como esses modelos evoluíram e funcionam. Um aviso: nem todas as técnicas que mencionarei a seguir são mutuamente exclusivas, e mesmo as mais antigas continuam sendo usadas hoje em dia em certos contextos. Essa é apenas uma visão geral.

A trajetória dos modelos de linguagem começa nos anos 50 e 60 com modelos baseados em regras, como o *Experimento Georgetown–IBM*[1] para tradução automática entre o inglês e o russo, e também *ELIZA, a robô terapeuta*[2] .

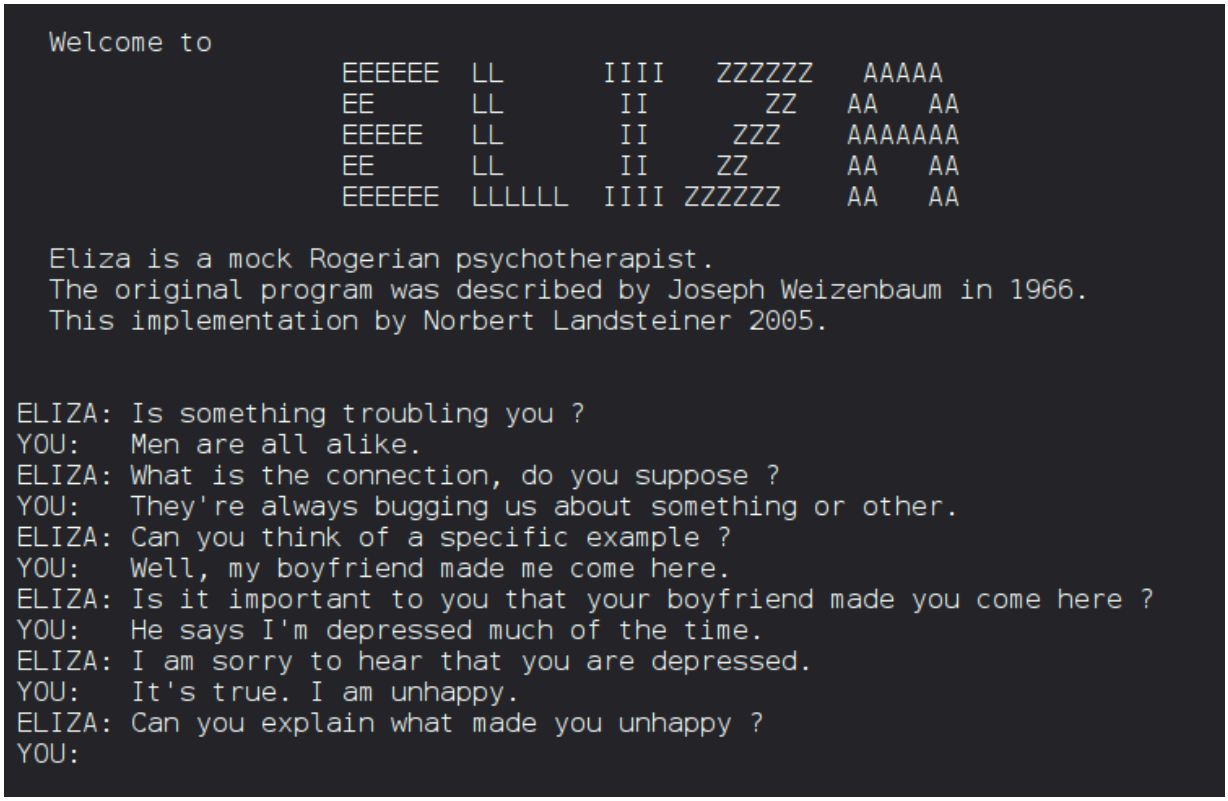

Figura 3.1: Exemplo de conversa com ELIZA, a robô terapeuta

Nas décadas seguintes, principalmente nos anos 80 e 90, as técnicas estatísticas ganharam mais força, como o famoso modelo de **N-grama** (*N-gram* em inglês)[3] .

N-grama

Essa técnica de modelagem probabilística parte da análise de grandes quantidades de texto, medindo a frequência que cada palavra (ou grupo de 'N' palavras) aparece e tentando adivinhar a próxima com base nas palavras imediatamente anteriores a ela.

Considere o texto de exemplo a seguir:

Oi, Giuliana. Tudo bem?

Viu, eu queria te avisar que eu não vou mais poder viajar com você na semana que vem. Infelizmente, surgiu um imprevisto ***de*** *família que eu não vou poder adiar. Eu vou ter que viajar para a cidade da minha família, porque a minha prima ficou muito doente e todo mundo da família está indo visitá-la.*

Você sabe o quanto eu gosto ***de*** *viajar e, especialmente, o quanto eu gosto* ***de*** *viajar com você, mas dessa vez realmente não vai dar.*

Uma outra opção é nós adiarmos a nossa data ***de*** *ida, se você não tiver nada contra. Dessa forma, daria tempo* ***de*** *eu voltar e a gente poderia viajar conforme o planejado, apenas uma ou duas semanas mais tarde.*

Me avisa o que acha dessa ideia. Beijos!

Vamos tentar calcular quais palavras têm maior probabilidade de aparecer depois da palavra "*de*". Primeiramente, devemos encontrar todas as palavras que aparecem depois da palavra "*de"* no texto acima e a frequência com a qual aparecem. Ao fazermos isso, chegamos a "família", "viajar" (2x), "ida" e "eu".

Em seguida, dividimos essa frequência que "*de*" aparece com cada uma delas pela frequência que a palavra "*de*" aparece em todo o texto ou conjunto de textos (neste caso, 5 vezes no total) para obter as probabilidades de cada um desses pares de palavras aparecer no texto. Por exemplo:

- **Prob("família" | "de") = Cont("de família") / Cont("de")**
  a probabilidade de termos a palavra "família" depois da palavra "de" é igual à quantidade de vezes que a expressão "de família" aparece no texto, dividida pela quantidade de vezes que a palavra "de" aparece no texto em geral

- **Prob("viajar" | "de") = Cont("de viajar") / Cont("de")**
  a probabilidade de termos a palavra "viajar" depois da palavra "de" é igual à quantidade de vezes que a expressão "de viajar" aparece no texto, dividida pela quantidade de vezes que a palavra "de" aparece no texto em geral

Então, para o texto acima, teríamos:

- **Prob("família" | "de") = 1/5**
- **Prob("viajar" | "de") = 2/5**

Isso quer dizer que haveria uma probabilidade de 20% de "família" ser a palavra a aparecer depois da palavra "*de*", e uma probabilidade de 40% de "*viajar*" ser essa palavra. Mas é claro, esse método não faz sentido para conjuntos tão pequenos de texto, precisamos de grandes conjuntos de treinamento.

Imagine, então, que tenhamos um conjunto gigantesco de textos para treinar o modelo, com todos os livros já escritos em língua portuguesa na história. Poderíamos então, simplesmente, calcular essa probabilidade para cada par de palavras que aparecer em nosso conjunto e pronto, teríamos o *ChatGPT*, certo? Na verdade, não.

O grande problema aqui é que estaríamos considerando apenas 1 palavra anterior como contexto para prever a próxima. Com um contexto tão pequeno, o modelo poderia gerar uma frase como "*Essa cidade é muito (lindo)*", pois ao calcular a probabilidade da palavra que virá logo após "muito", ele teria apenas a própria palavra anterior ("muito") como contexto, e não saberia, assim, que a palavra a ser prevista teria que estar no gênero feminino, por se relacionar à palavra "cidade".

Ah, e só pra esclarecer o nome da técnica "*N-grama*", esse acima foi um exemplo de *2-grama* ou *bigrama*, que é quando há uma sequência de 2 palavras — nesse caso, a de contexto e a que será prevista. Um modelo de *1-grama* ou *unigrama*, por exemplo, tentaria calcular a próxima palavra usando apenas suas frequências, sem nenhuma palavra anterior de contexto.

Considerando um símbolo para indicar o início de uma frase e outro para indicar o seu término, para calcular a probabilidade aproximada da frase "*Essa cidade é muito linda*" usando um modelo *2-grama*, faríamos:

- probabilidade de "símbolo de início de frase" e da palavra "Essa"
- probabilidade da palavra "Essa" seguida pela palavra "cidade"
- probabilidade da palavra "cidade" seguida pela palavra "é"
- probabilidade da palavra "é" seguida pela palavra "muito"
- probabilidade da palavra "muito" seguida pela palavra "linda"
- probabilidade da palavra "linda" seguida por "símbolo de fim de frase"

E então, multiplicar os valores resultantes.

Agora, você pode me perguntar: "*E se nós aumentarmos esse contexto? Se tentarmos usar as últimas 5, 10 ou 30 palavras como contexto para tentar prever a próxima?*"

Bom, poderíamos ter, por exemplo, um modelo de 7-grama com o contexto "*A melhor coisa da vida é (...)*", que calcularia a probabilidade da próxima palavra com base em toda sua base de dados colossal de treinamento, e provavelmente teria como resultado algo como "viajar", "aproveitar", "comer" etc. Mas e se tivéssemos uma frase como "*O rapaz que foi com a minha irmã na festa ontem estava (...)*"? Um modelo de 7-grama veria como contexto apenas o trecho "*minha irmã na festa ontem estava (...)*" e poderia, portanto, prever uma palavra como "*charmosa*" para ser a próxima, por estar enxergando apenas o contexto que cita "*minha irmã*", e não a parte que menciona o rapaz.

Percebe como o contexto anterior é extremamente importante em um modelo de linguagem? Porém, quanto maior o valor de *N* de um modelo de *N-grama*, mais difícil será encontrar aquela sequência exata de palavras no conjunto de dados de treinamento. Modelos N-grama também sofrem do problema de escassez de dados: se uma certa sequência de palavras não apareceu no texto de treinamento,

o modelo terá dificuldade em predizer a próxima palavra, mesmo que faça sentido gramaticalmente. Além disso, a quantidade de possíveis *N-gramas* aumenta exponencialmente com o aumento de seu *N* e, assim, chegaríamos rapidamente a um ponto em que o treinamento e uso do modelo não seria factível em termos de tempo e/ou de memória computacional.

Os modelos de *N-grama* têm sido muito usados há décadas por serem fáceis de entender e implementar, darem bons resultados e serem aplicáveis a diversos tipos de ferramentas que usamos diariamente, como corretores ortográficos, sistemas de previsão de palavras em teclados virtuais, tradutores automáticos, detectores de plágio, sistemas de reconhecimento de fala e assistentes virtuais.

Word Embeddings e Redes Neurais

As décadas de 2000 e, principalmente, de 2010 viram um grande ressurgimento das redes neurais em várias áreas, um conceito não tão novo, mas que se beneficiou do aumento da capacidade de processamento de computadores e *GPUs* e de seu barateamento.

O modelo de linguagem proposto por Yoshua Bengio *et al.* no início dos anos 2000[4] , muitas vezes referido como o primeiro modelo de rede neural especificamente projetado para processamento de linguagem, foi um marco significativo na evolução dos modelos de linguagem e no campo de Processamento de Linguagem Natural. Sua grande inovação, além da aplicação de redes neurais, foi o uso de representações distribuídas de palavras que compreenderiam a proximidade semântica entre elas.

Imagine a palavra "**avião**". Se nós quiséssemos representá-la em uma dimensão que reconhecesse "meios de transporte", essa palavra ficaria próxima a outras como "carro", "trem", "ônibus" etc. Porém, as palavras, no geral, não existem em apenas uma única dimensão. Poderia haver uma dimensão que reconhecesse

"atividades", e aí, a palavra "avião" ficaria próxima de palavras como "voar", "aterrissar", "abastecer" etc. Porém, no geral, essa palavra ficaria bem distante de outras como "acariciar", "focinho" ou "xícara", por não ocorrerem quase nunca em um mesmo contexto (a menos, é claro, que você esteja lendo um poema surrealista sobre uma xícara que acaricia um avião com seu focinho. Mas, para o bem da sanidade do modelo, vamos assumir que esses textos são a minoria).

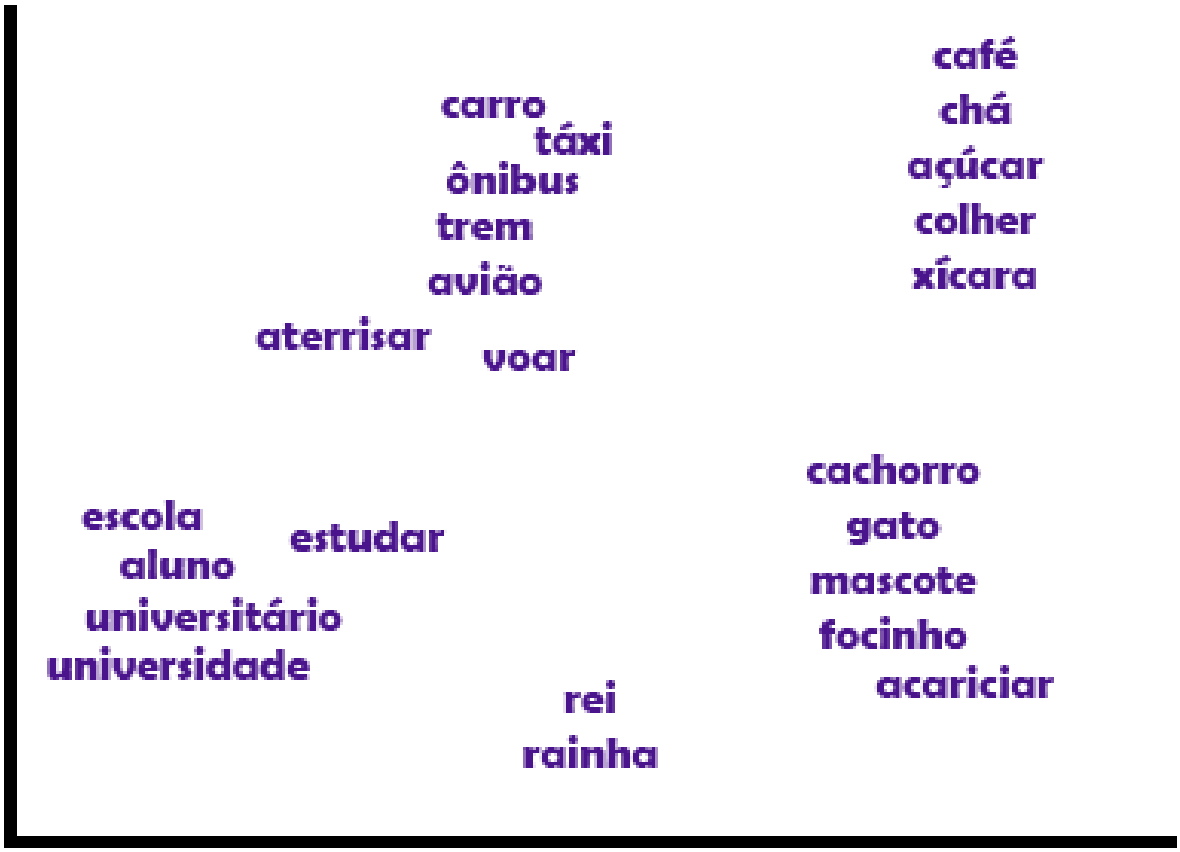

Figura 3.2: Exemplo de word embeddings em 2D

Cada palavra ("*word*" em inglês) seria associada a um vetor numérico (conhecido aqui como "*embedding*") em um espaço dimensional, e as palavras que fossem mais semelhantes semanticamente teriam vetores mais próximos uns dos outros nesse espaço geométrico. Em aplicações reais, eles geralmente têm

centenas de dimensões, e quanto mais dimensões houver, melhor seria a nossa compreensão sobre essa palavra e sua relação com outras em diferentes contextos. Isso é conhecido no mundo de PLN como "***word embedding***"[5] .

Você pode ver a seguir um exemplo de *embedding* para a palavra "avião":

	avião
voar	0,95
trem	0,78
ônibus	0,76
...	
focinho	0,015
xícara	0,09
abastecer	0,61

Figura 3.3: Exemplo de embedding da palavra "avião"

Na prática, um vetor de *embeddings* não teria essa relação tão clara. A seguir, podemos ver um vetor de *embeddings* de 20 dimensões, com uma precisão de 16 bits por valor:

```
embeddings = [0.1785888671875, -0.28759765625, 0.2374267578125, -0.26171875, 0.630859375,
              -0.8046875, 0.19287109375, -0.059478759765625, 0.278076171875, -0.70361328125,
              0.433837890625, -0.325927734375, 0.340087890625, 0.22412109375, -0.68701171875,
              -0.92431640625, 0.98291015625, 0.712890625, 0.1849365234375, -0.06494140625]
```

Figura 3.4: Exemplo de embedding de 20 dimensões e 16 bits

Porém, na vida real, os vetores de *embeddings* terão 384, 768, 1024 ou mais dimensões. É praticamente impossível visualizar.

Existem vários modos de treinar um modelo para gerar esses vetores de *embeddings*, mas um dos mais revolucionários foi proposto em 2013 em um artigo científico do *Google* escrito por *Mikolov et al.*[6] , o **Word2vec**. Trata-se de um algoritmo de rede neural rasa (lembre-se que redes neurais profundas são aquelas com mais de uma camada oculta) que se baseia em duas técnicas principais: *Continuous Bag of Words* (*CBOW*) e *Skip-Gram*.

No *CBOW*, treinaríamos o modelo para prever uma palavra considerando as palavras no seu entorno. Por exemplo, dada a frase "*Meu _____ latiu muito durante o passeio*", o modelo teria que tentar prever a palavra "cachorro" com base nesse contexto ao seu redor.

Já no *Skip-Gram*, é exatamente o contrário. Usando a mesma frase, treinaríamos o modelo para prever o contexto dada uma palavra central. Por exemplo, dada a palavra "*cachorro*", a tarefa do modelo seria prever o contexto a seu redor, no nosso caso: "meu", "latiu", "muito", "durante", "o" e "passeio".

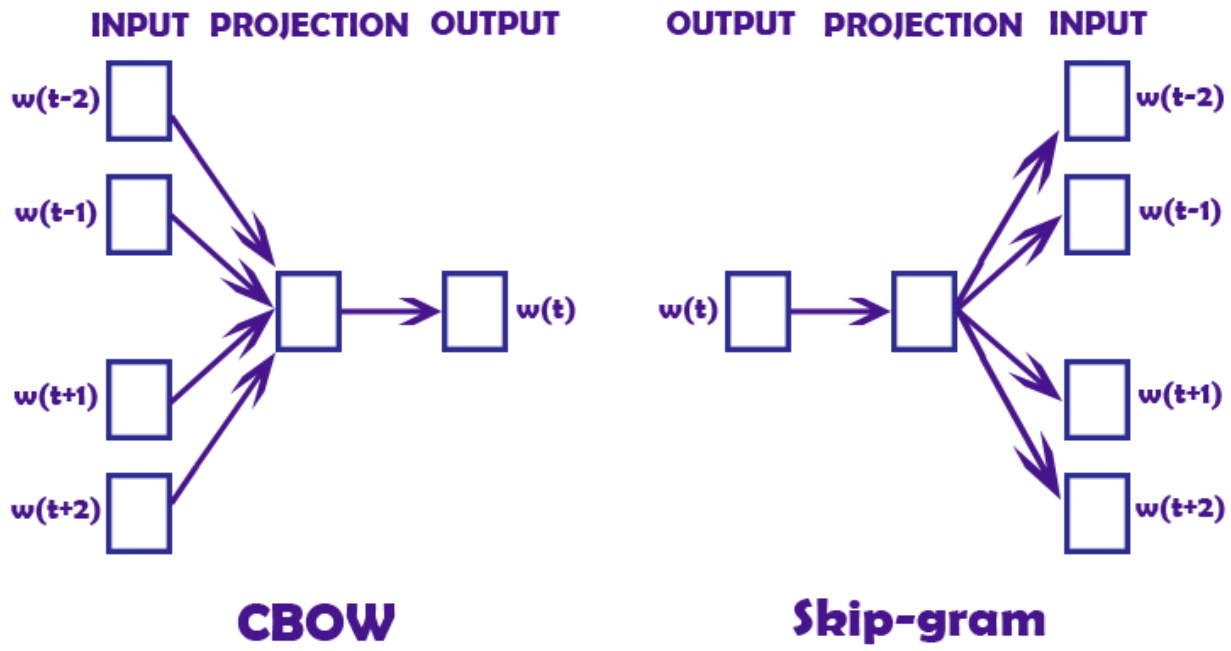

Figura 3.5: Modelos CBOW e Skip-Gram propostos por Mikolov et al.

Essa técnica tornou possível que os modelos aprendessem *word embeddings* que capturassem seu contexto semântico e relações com outras palavras. Outro fato superinteressante dessa técnica é que também se percebeu que, no espaço geométrico onde esses *embeddings* estivessem representados, seria possível fazer até mesmo operações matemáticas "semânticas" com eles. Por exemplo, se pegássemos o vetor da palavra "rei", subtraíssemos dele o vetor da palavra "homem" e adicionássemos o vetor da palavra "mulher", obteríamos como resultado um vetor muito próximo do vetor da palavra "rainha" (diz a lenda que o vetor de "Fabrício Carraro" somado ao vetor da palavra "open bar" resulta no vetor da palavra "arrependimento").

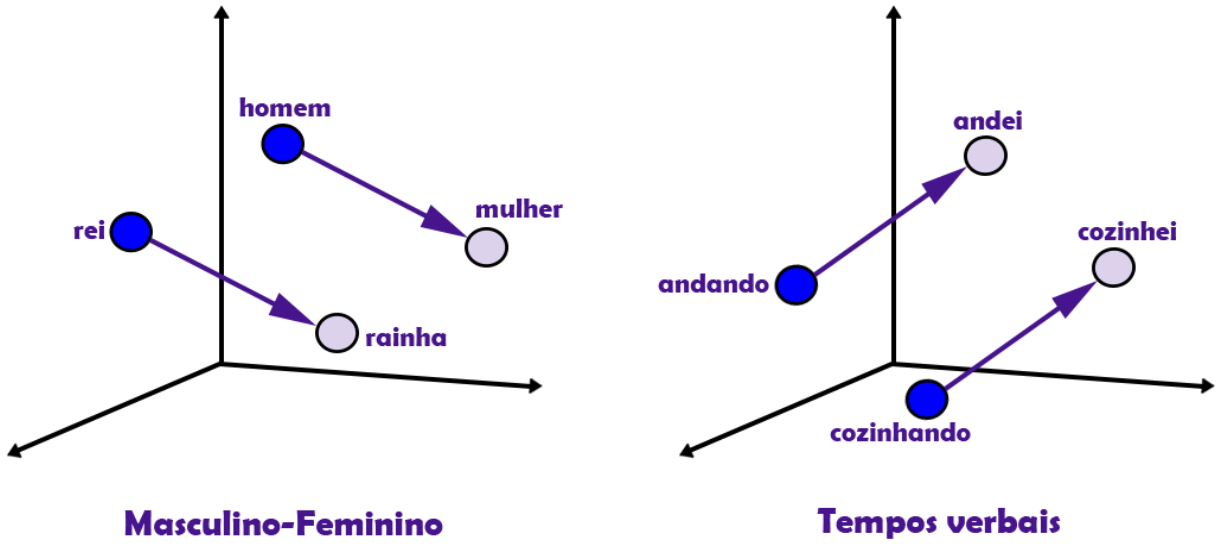

Figura 3.6: Exemplo de relações capturadas pelo modelo Word2vec

E foi nesse período que o foco das pesquisas de PLN se voltou para o *Deep Learning* e as Redes Neurais Recorrentes.

A Era das Redes Neurais Profundas

Não falei que o conteúdo do capítulo anterior seria importante? Você se lembra dos tipos de redes neurais que aprendemos no capítulo 2? Um deles foi o conceito de Redes Neurais Recorrentes (*Recurrent Neural Networks* ou *RNN* em inglês)[7] que, recapitulando, são redes neurais profundas que permitem "*loops*", ou seja, diferentemente de *Redes Neurais Feedforward* (onde o resultado de um neurônio só é transmitido para neurônios de camadas subsequentes), em Redes Neurais Recorrentes, a saída de um nó pode voltar para a entrada do próprio nó. Por esse motivo, elas conseguem levar em consideração o que viram anteriormente, ocasionando um tipo de "memória" — algo muito útil quando precisamos nos lembrar do contexto de uma frase, não é mesmo?

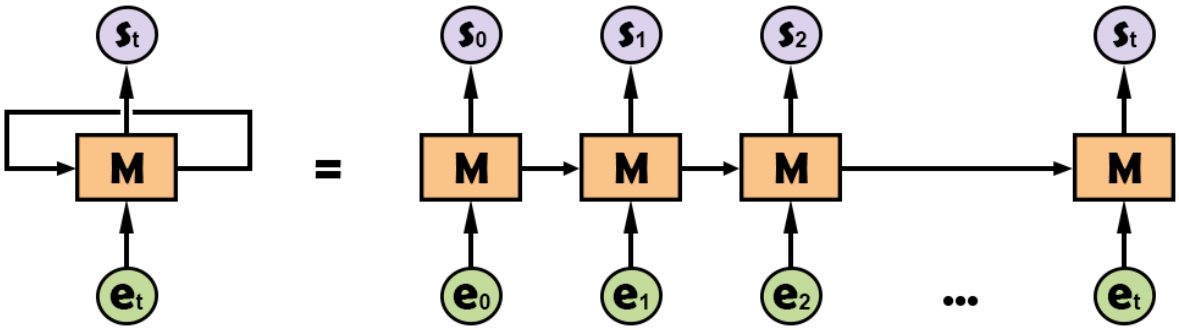

Figura 3.7: Visão de uma Rede Neural Recorrente ao longo do tempo de execução

Aplicando o conceito de *RNN* para modelos de linguagem, seu funcionamento se dá de tal forma: uma sequência de palavras (já convertidas em vetores numéricos de *word embeddings*) é processada em ordem, uma após a outra, e isso gera um "estado oculto", que é basicamente um vetor numérico que atua como uma representação das informações mais relevantes da sequência de palavras vistas até então. Ou seja, esse estado oculto não armazena exatamente as palavras vistas até então, mas uma representação codificada delas em forma numérica que resume o contexto e que será usada na hora de gerar a próxima palavra.

Vamos supor que tenhamos a frase "*O astronauta quer pisar na (...)*". A nossa *RNN* iria primeiramente receber o *word embedding* da palavra "O" e atualizar o estado oculto da rede com informações sobre a palavra "O". A seguir, processaria o *word embedding* da palavra "astronauta" e atualizaria o estado oculto com base nas informações da palavra "astronauta", levando em consideração o contexto anterior (que inclui a palavra "O") — ou seja, o estado oculto agora conteria informações codificadas sobre as palavras "O" e "astronauta". Esse processo continuaria à medida que a *RNN* processasse cada palavra subsequente na frase, sempre atualizando o estado oculto a cada nova palavra processada. Assim, a *RNN* vai construindo uma representação numérica cada vez mais rica da frase conforme processa cada palavra.

Seguindo essa lógica, ao chegar na palavra "na", a entrada do modelo seria o *word embedding* dessa própria palavra e também o

estado oculto com a representação de todas as palavras vistas até então — lembrando que o modelo não "vê" as palavras "O", "astronauta", "quer" etc. literalmente, mas sim a informação semântica dessas palavras, que foi codificada numericamente nesse estado oculto ao longo das atualizações. Assim, dependendo de seu treinamento, é provável que ao final do processo o modelo conseguiria prever algo como "Lua" para a próxima palavra da frase.

E agora vem a pergunta: "*Um modelo baseado em RNN é melhor do que um modelo baseado em N-grama?*". E a resposta, como em muitos casos, é: depende. Como vimos, um modelo *N-grama* consegue ter como contexto apenas um número de palavras de *N - 1* (ou seja, o modelo 7-grama que vimos tem 6 palavras de contexto), e temos que nos contentar com isso (o modelo não consegue usar como contexto nada anterior a isso). Outra opção é aumentar o valor de *N*, o que aumentaria esse contexto, mas como vimos, também aumentaria exponencialmente o tamanho do modelo e poderia torná-lo infactível em termos de tempo ou memória computacional.

Já um modelo de linguagem baseado em uma *RNN* tem a capacidade de incluir informações de contexto muito mais longas e sem crescer exponencialmente, pois o estado oculto não armazena as palavras em si, mas sim uma representação codificada numericamente em um vetor. Porém, isso também pode causar um problema. Esse vetor tem um tamanho fixo e, para sequências de palavras mais longas, existe um problema chamado de "**dissipação do gradiente**" (*vanishing gradient* em inglês)[8] , onde a influência das palavras muito anteriores na sequência se torna cada vez mais fraca e pode se tornar até mesmo insignificante em etapas posteriores. É como aquele amigo que começa a contar uma piada e se perde no meio do caminho. Isso dificulta a capacidade de a *RNN* aprender relações entre palavras distantes.

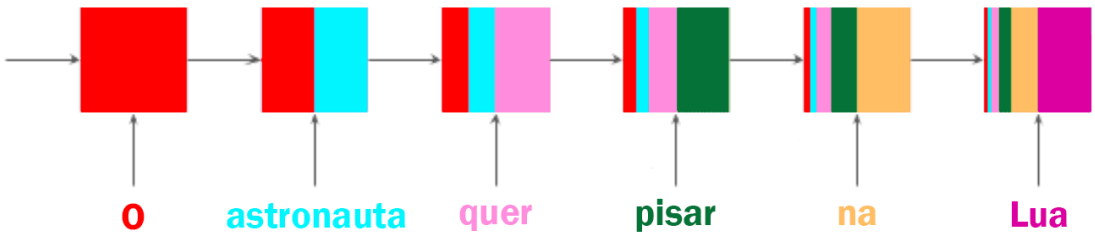

Figura 3.8: Exemplo da dissipação do gradiente

Devido a esse problema, outras abordagens passaram a ser utilizadas. Uma das mais bem-sucedidas é a de *LSTM* (*Long Short-Term Memory*, traduzindo, "Redes de Memória Longa de Curto Prazo")[9] , também já citada no capítulo anterior. Recapitulando, uma *LSTM* é um tipo especial de *RNN* que pode manter sua memória de curto prazo por um longo período de tempo. Ela tem estruturas chamadas de "portas" (*gates*), que definem quais informações serão lembradas, descartadas ou passadas adiante.

As *RNNs* e *LSTMs* têm uma outra desvantagem. Devido ao fato de que processam as palavras na sequência, uma após a outra, elas não podem ser treinadas de maneira paralela — ou seja, para processar a palavra *X*, você precisa ter processado a palavra *X - 1* e ter a representação de seu estado oculto; para processar a palavra *X - 1*, você precisa ter processado a palavra *X - 2* e ter a representação de seu estado oculto; e assim por diante, o que pode causar um gargalo no treinamento de modelos de linguagem e a necessidade de um enorme poder computacional, dependendo do tipo de modelo a ser treinado.

Dessa forma, podemos dizer de maneira simples que modelos baseados em *N-grama* enxergam um contexto bem mais limitado, mas o fazem de maneira clara, por conseguirem ver as palavras literalmente; já modelos baseados em *RNN* conseguem enxergar um contexto bastante maior, o que permite modelos de linguagem exponencialmente melhores, mas não veem esse contexto de maneira clara, apenas uma imagem borrada dele (o estado oculto,

condensado em um vetor numérico). Ou seja, a escolha era entre um modelo com a memória de um peixinho dourado, mas com visão de águia (*N-grama*), e outro com a memória de um elefante, mas que precisa de óculos com um grau bem alto (*RNN*). Nenhum dos dois era o ideal.

Atenção!

O ano de 2014 foi chave para algumas das inovações mais relevantes no mundo de PLN. Ilya Sutskever e seu time no *Google* propuseram o modelo *Sequence to Sequence* ("Sequência a sequência", em português)[10] , a fim de converter uma sequência de tipo A para uma outra sequência de tipo B. Uma aplicação óbvia é em tradução automática, convertendo uma sequência de palavras de inglês para português, por exemplo.

O modelo proposto utilizaria uma arquitetura de rede neural *Encoder-Decoder*. Explicando melhor, o *Encoder* ("codificador", em português) pegaria a frase de entrada, nesse caso em inglês, converteria cada palavra para seu vetor numérico de *word embedding* e, a partir dessa sequência de *word embeddings*, geraria um "vetor resumo" que captura o sentido geral da frase. Imagine isso como um intérprete ouvindo uma frase, compreendendo o significado, e criando em sua mente uma imagem do que foi dito. Em seguida, o *Decoder* ("decodificador", em português) pegaria esse "vetor resumo" numérico de representação e o utilizaria para gerar a frase de saída em português. Isso seria semelhante ao intérprete falando em voz alta a frase em outro idioma, usando a compreensão que ele havia formado em sua mente.

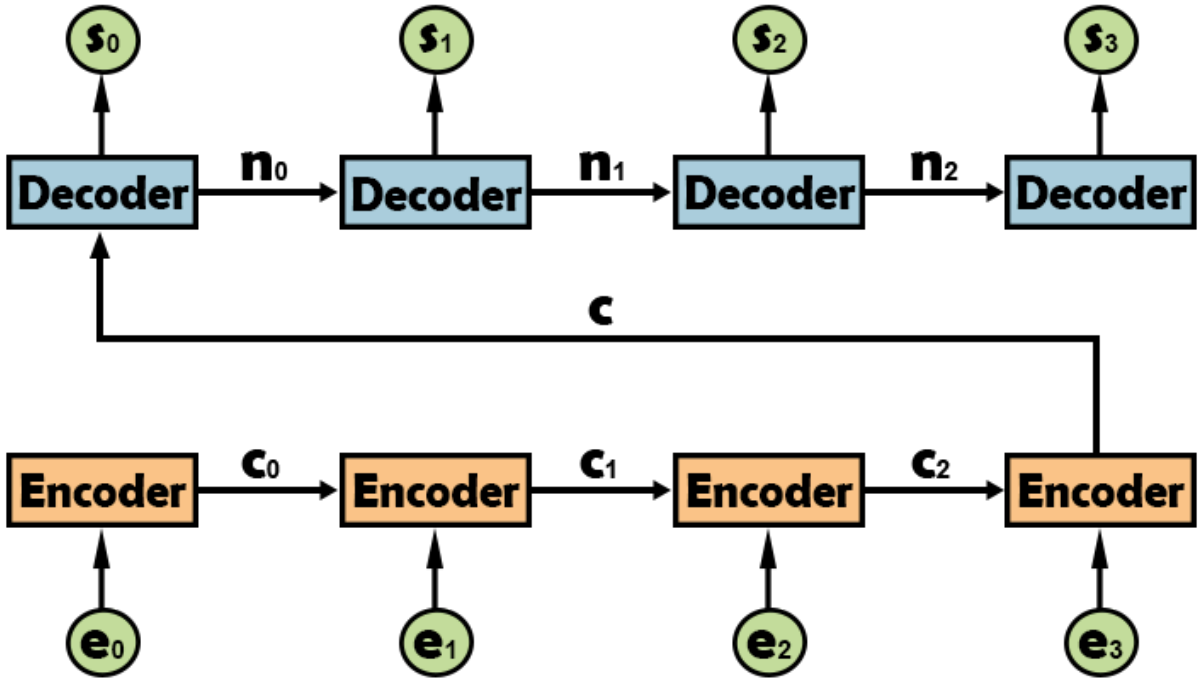

Figura 3.9: Visão de uma arquitetura Encoder-Decoder simples

A arquitetura da rede neural *Encoder-Decoder* proposta por Sutskever teria uma rede neural *LSTM* multicamadas no *Encoder* para processar uma frase (em inglês, por exemplo) palavra por palavra e criar seu vetor de representação ("**c**", na imagem anterior), e outra rede neural *LSTM* no *Decoder* para prever a saída (em português, por exemplo) com base no vetor gerado. Porém, essa proposta não é uma restrição. Poderíamos ter uma arquitetura *Encoder-Decoder* com redes *RNN*, *LSTM* ou *GRU* como o *Encoder* e/ou o *Decoder*, por exemplo.

Apesar de ter obtido bons resultados, um dos problemas do modelo original era que esse vetor de representação ("**c**", na imagem) com toda a sequência de entradas do *Encoder* tinha um tamanho fixo, o que poderia causar perdas de informação de contexto sobre a frase sendo traduzida, como vimos anteriormente. Um vetor de tamanho fixo implicaria que toda a informação da frase de entrada, curta ou longa, simples ou complexa, precisaria caber nessas mesmas "N"

dimensões. Isso causava um grande gargalo para frases mais longas ou complexas.

A fim de mitigar esse problema, surge o famoso **mecanismo de atenção**, proposto também em 2014 por Dzmitry Bahdanau, Kyunghyun Cho e Yoshua Bengio[11] . A ideia é simples, mas poderosa: em vez de usar sempre o mesmo "vetor resumo" de tamanho fixo da frase de entrada para gerar a tradução final, esse mecanismo permite que o *Decoder* tenha um vetor de representação dinâmico, que será recalculado para cada palavra nova a ser gerada na saída. Com isso, o modelo consegue analisar quais partes da frase de entrada são mais relevantes para prever a próxima palavra na frase de saída, de modo que ele possa se concentrar nelas ao calcular as probabilidades da próxima palavra.

Funciona mais ou menos como um tradutor que, ao escrever a tradução de um artigo, pode voltar e reler trechos específicos do texto original sempre que precisar, e não apenas confiar em uma lembrança condensada.

Por exemplo, na tradução da frase "*The book is on the table*" do inglês para o português, o modelo pode não se concentrar tanto na palavra "*the*" em inglês, mas dar mais atenção à palavra "*book*", por entender que essa palavra carrega mais significado na hora de prever o verbo que virá na sequência ("*the book is*" - "o livro está", mas "*the books are*" - "os livros estão").

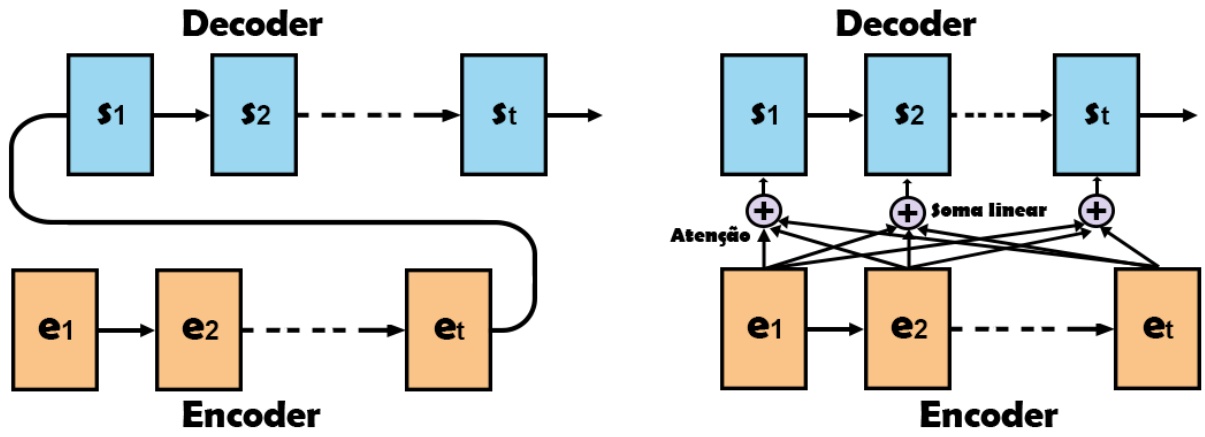

Figura 3.10: No lado esquerdo, visão geral de uma arquitetura Encoder-Decoder simples; no lado direito, Encoder-Decoder com atenção

Como podemos ver, antes do mecanismo de atenção, o modelo faria uma análise completa do texto em inglês (entrada) no *Encoder*, geraria um vetor com toda essa informação condensada e, então, a enviaria para o *Decoder* para tentar produzir a tradução completa de uma só vez para o português (saída).

Com o mecanismo de atenção, esse método é aprimorado. O modelo analisa apenas uma parte da entrada em inglês de cada vez e, dessa forma, pode "prestar atenção" a várias partes específicas da frase enquanto a traduz. Ele atribui um "peso" para cada uma dessas partes da frase, e quanto maior o peso atribuído a uma determinada palavra ou trecho, mais "atenção" o modelo prestará a ela ao produzir a próxima palavra na tradução. Ao final, há uma soma ponderada desses pesos, onde cada palavra na entrada recebe um valor de pontuação com base nos pesos alocados nas etapas anteriores. Isso será usado para produzir a saída final.

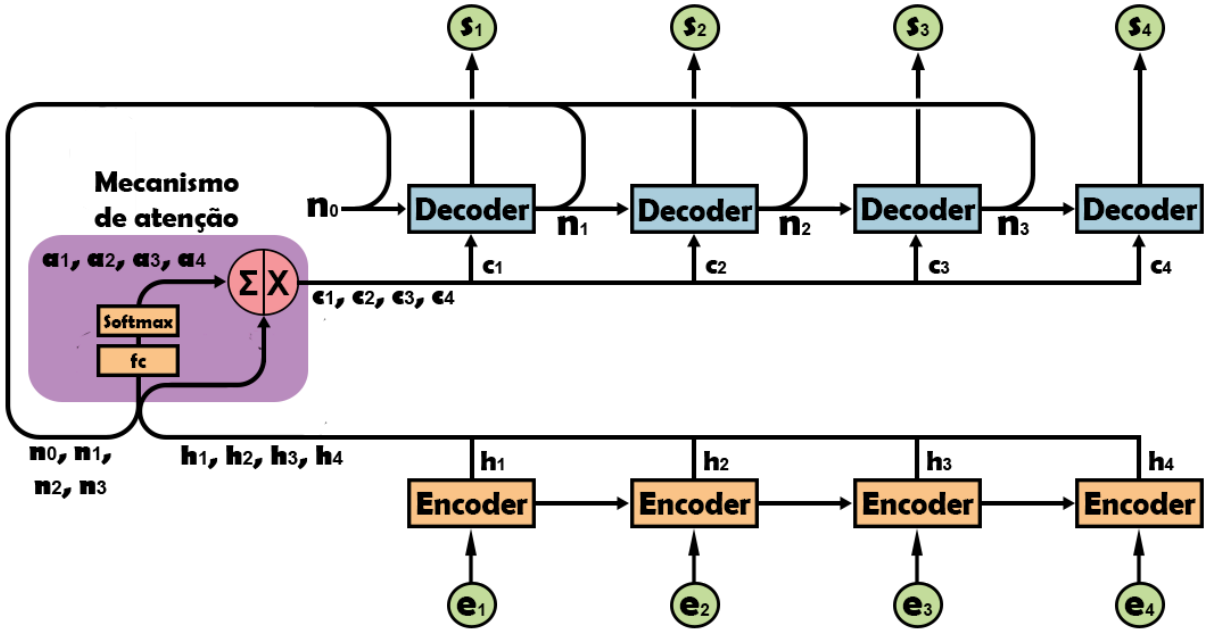

Figura 3.11: Visão geral de uma arquitetura Encoder-Decoder com atenção

Esse mecanismo de atenção é ainda mais útil na tradução de idiomas com estruturas gramaticais diferentes, onde a ordem das palavras em um deles pode não corresponder diretamente à ordem no outro, como é o caso do exemplo citado anteriormente sobre a língua turca, onde os verbos vão sempre no final.

Essas inovações em PLN tiveram tanto sucesso que, em 2016, o *Google* anunciou que havia substituído seu algoritmo de tradução do *Google Tradutor* de um modelo baseado em frases para um modelo baseado em redes neurais com o mecanismo de atenção[12] .

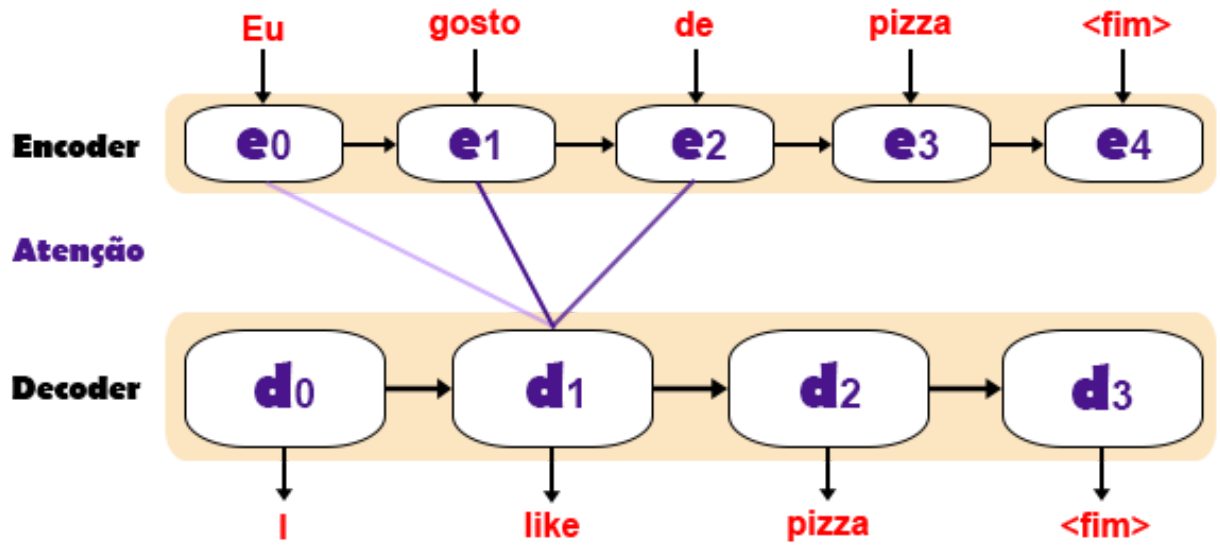

Figura 3.12: Demonstração do funcionamento da nova arquitetura do Google Tradutor (2016) usando redes neurais e atenção

Pouco depois do artigo sobre o mecanismo de atenção, outros estudos começaram a ser feitos sobre o seu possível uso na área de análise de imagens. Um deles, proposto por *Kelvin Xu et al.* em 2015[13] , apresentava um modelo usando Redes Neurais Convolucionais baseado no mecanismo de atenção para descrever o conteúdo de imagens. Você pode ver claramente na imagem a seguir como o modelo "presta atenção" às partes corretas da imagem que correspondem à palavra em análise naquele momento.

Figura 3.13: Exemplo da aplicação do mecanismo de atenção para descrição de imagens

Mas as inovações do time de pesquisa do *Google* não pararam por aí.

Transformers

Não, esse tópico não é sobre a série de filmes de Optimus Prime, Bumblebee e Megatron.

Conforme mencionado anteriormente, uma das desvantagens de redes neurais *RNN* e *LSTM* era o fato de que elas funcionam de maneira sequencial — para processar a palavra *X*, você precisa ter processado a palavra *X - 1* e ter a representação de seu estado oculto, e assim por diante — o que gera um gargalo no processamento. E se pudéssemos fazer todo esse processamento de maneira paralela? Não seria fantástico?

Para resolver esse problema, tudo o que você precisa é atenção — mais especificamente, do mecanismo de atenção. Esse é exatamente o título de um artigo científico publicado por Ashish Vaswani e seu time no *Google* em 2017 ("*Attention Is All You Need*"[14]) que revolucionou o mundo da Inteligência Artificial e dos modelos de linguagem através da apresentação da arquitetura de *Transformers*.

Essa nova arquitetura proposta também foi desenhada com uma base *Encoder-Decoder*, mas sem utilizar Redes Neurais Recorrentes ou *LSTMs*, e tendo como seu conceito-chave o mecanismo de atenção que vimos anteriormente. Essa estrutura permite o funcionamento de maneira paralela e elimina o gargalo que das *RNNs* e *LSTMs*. Ela apresentou também dois novos mecanismos: ***self-attention*** ("autoatenção", em português) e ***positional encoding*** ("codificação da posição", em português) para ajudar a rede a se concentrar nas palavras mais relevantes durante o processamento e sua relação com outras palavras.

Para explicar o mecanismo de *self-attention*, considere a frase: "*A galinha não atravessou a rua porque ela estava cansada*". A que se refere o pronome "ela" nessa frase, à "galinha" ou à "rua"? Em

geral, você poderia instintivamente "prestar atenção" em outras partes da frase e buscar pistas sobre quem é "ela" olhando para trás no texto. É claro que, para um ser humano com vivências no mundo real, seria fácil deduzir que o pronome em questão se refere à "galinha", mas um algoritmo pode não ter tanta facilidade.

O mecanismo de atenção "padrão" que havíamos visto anteriormente obtinha o vetor de representação que vinha do resultado do *Encoder* e, na sequência, durante a fase do *Decoder*, conseguia prestar atenção a diferentes partes da entrada para cada palavra da saída. Já esse novo mecanismo de *self-attention* em uma arquitetura de *Transformers* traz duas vantagens principais.

Primeiro, ele permite muito mais paralelismo. Em vez de processar palavra por palavra de forma sequencial, o modelo pode analisar todas ao mesmo tempo. Cada palavra da sequência observa todas as outras palavras da mesma sequência e atribui uma pontuação a cada uma delas, para entender como estão relacionadas e em quais deve prestar mais atenção. Em termos técnicos, podemos dizer que o mecanismo de *self-attention* calcula o peso da importância relativa dessa palavra e a relação entre todas as palavras em uma determinada frase. Dessa forma, é como se cada palavra tivesse uma visão panorâmica da frase inteira, conseguindo identificar quais outras palavras são mais importantes para compor seu próprio significado.

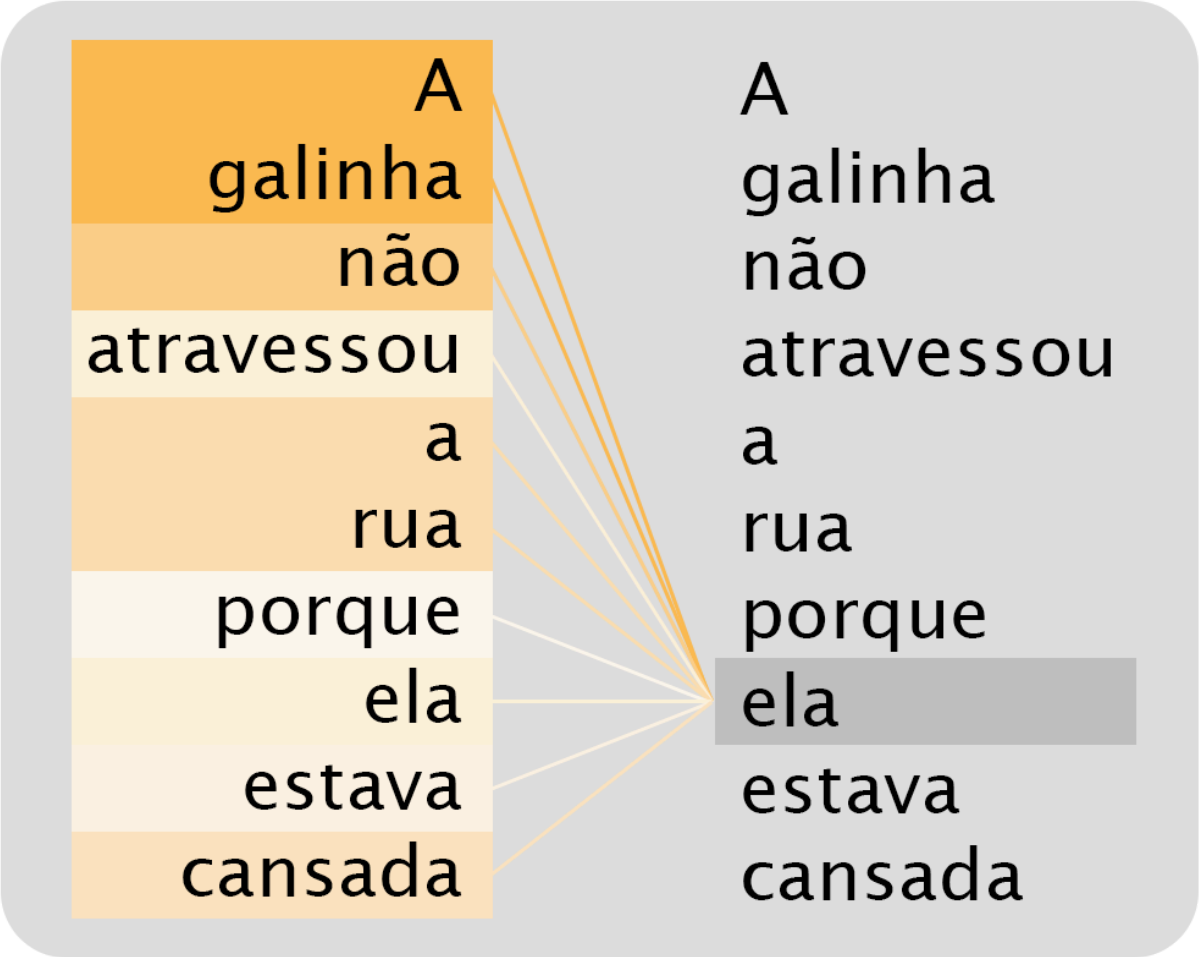

Figura 3.14: Exemplo do funcionamento da self-attention

Segundo, além dessa análise paralela, os *Transformers* introduziram outro avanço importante: o mecanismo de ***multi-head attention*** ("atenção multicabeças" na tradução livre em português). Em vez de calcular uma única "atenção" entre as palavras da frase, o modelo calcula várias "atenções" independentes ao mesmo tempo, cada uma chamada de "cabeça". Enquanto uma cabeça de atenção pode se especializar em identificar quem é o sujeito e qual é o verbo da frase, outra pode se concentrar nas ligações entre adjetivos e substantivos, ou mesmo em saber as relações de pronomes e substantivos (como no exemplo do pronome "ela" estar se referindo a "galinha" ou "rua" na frase que já vimos anteriormente). Ao final, todas essas "visões" são combinadas para formar uma compreensão

mais rica e completa do contexto, que consegue capturar nuances mais sutis da linguagem.

Como todas as "atenções" são calculadas em paralelo dentro de cada etapa do processamento, e não de forma sequencial, o modelo não precisa esperar os passos anteriores serem completados antes de iniciar o próximo. Isso torna o treinamento dos modelos muito mais rápido do que em abordagens anteriores, como as redes neurais recorrentes, e possibilita a criação de modelos de linguagem muito maiores e mais poderosos.

No entanto, ao processar as palavras de forma independente e paralela, sem uma Rede Neural Recorrente ou Convolucional para analisá-las em sequência, como poderemos saber a posição de cada uma delas na frase? Para isso, foi introduzido o *positional encoding* ("codificação da posição", em português). Ele é um vetor que é adicionado a cada *embedding* da entrada e funciona como uma espécie de carimbo que indica a ordem e a distância entre elas. Isso é essencial, porque o significado das frases pode mudar completamente dependendo da posição das palavras, como nas frases "*O gato caça o rato*" e "*O rato caça o gato*".

Agora, uma parte um pouco mais técnica e detalhada.

Conforme mencionado anteriormente, a arquitetura de *Transformers* tem a estrutura base de *Encoder-Decoder*, que pode ser vista na imagem a seguir:

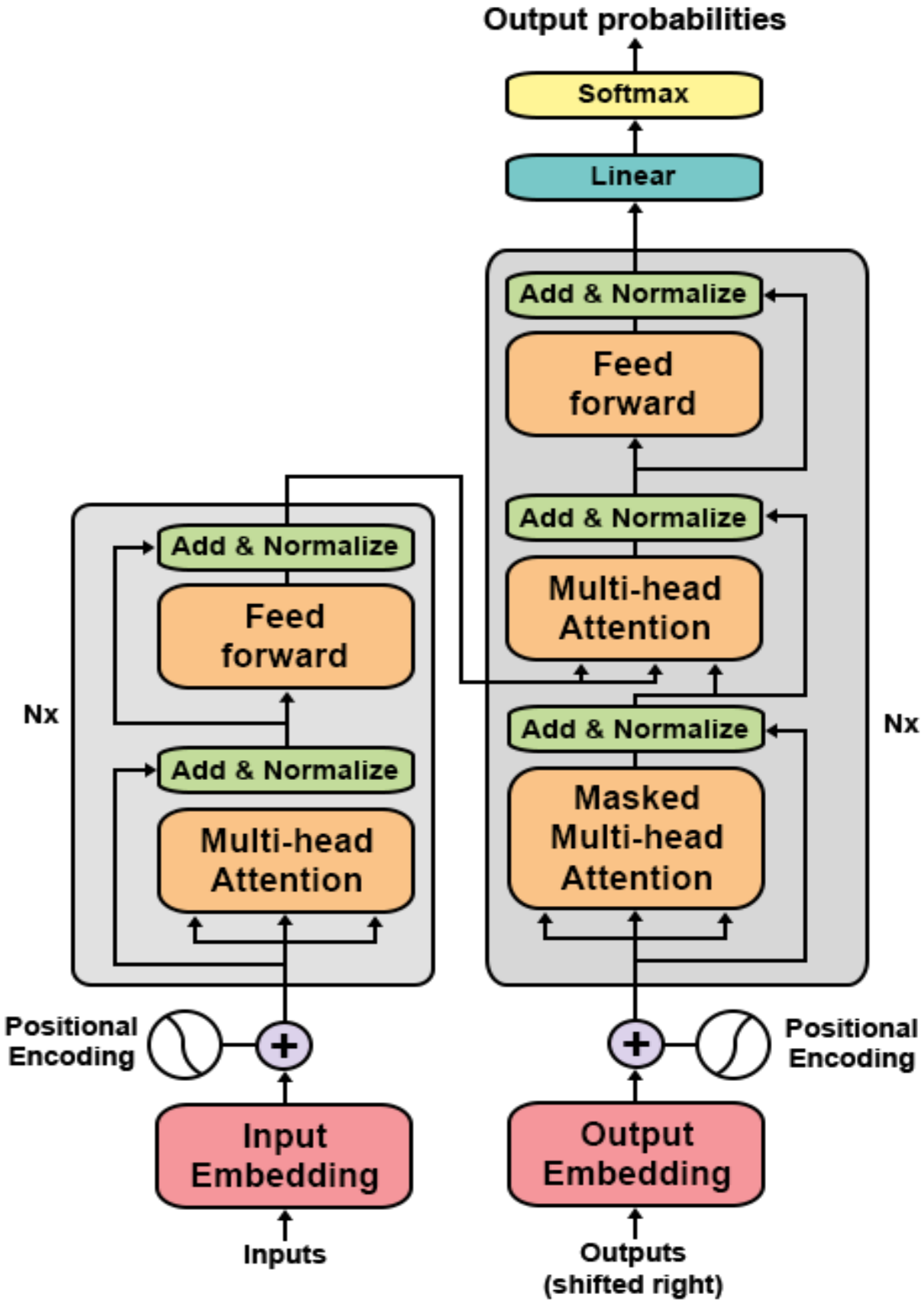
Output probabilities
Softmax
Linear
Add & Normalize
Feed forward
Add & Normalize
Multi-head Attention
Add & Normalize
Masked Multi-head Attention
Nx
Add & Normalize
Feed forward
Add & Normalize
Multi-head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Figura 3.15: Arquitetura de um modelo Transformer

O *Encoder* é a estrutura do lado esquerdo, que recebe a entrada, e o *Decoder* a estrutura do lado direito, que produz a saída. Olhando mais a fundo, porém, a estrutura de *Encoder* proposta no artigo de Vaswani é composta de uma pilha de seis *Encoders*, e o mesmo é verdadeiro para a estrutura de *Decoders*:

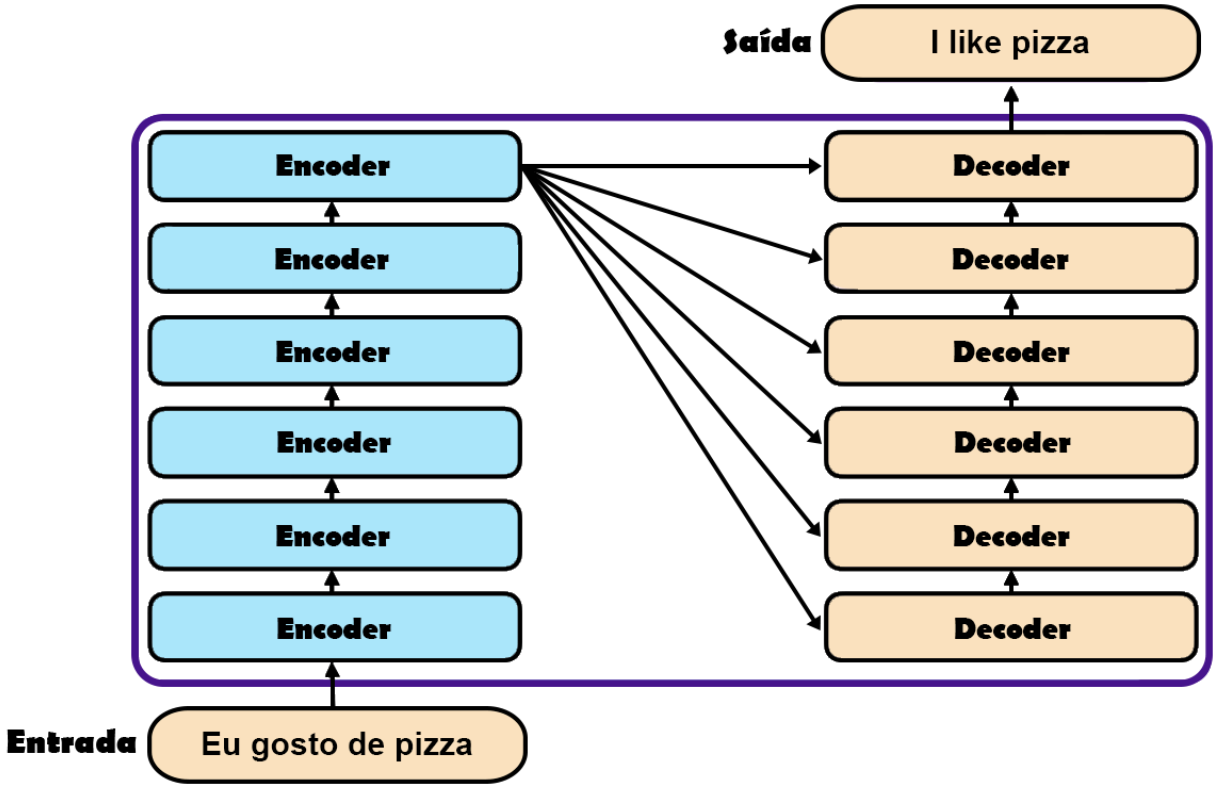

Figura 3.16: Arquitetura simplificada de um modelo Transformer por dentro

Essa pilha de *Encoders* recebe, além das entradas, seus vetores de *positional encoding*. Todos os seis *Encoders* são idênticos e são compostos de uma primeira camada de *self-attention*, cuja saída é normalizada e passada para a próxima camada: uma Rede Neural *Feedforward* (falamos sobre elas no capítulo 2), cujo resultado também é normalizado e passado adiante para os *Decoders*. As seis camadas do *Encoder* aplicam as mesmas operações a todas as palavras da sequência de entrada, mas cada camada tem pesos diferentes.

Imagine que cada *Encoder* seja um revisor literário. O texto (a sequência de palavras + o "marcador de página" do positional encoding) passa pelo primeiro revisor, que lê o capítulo inteiro, faz anotações nos pontos importantes e repassa o manuscrito ao seguinte. O segundo revisor recebe o texto já anotado, acrescenta suas próprias observações e o entrega ao terceiro, e assim por diante. Temos seis revisores no total, cada um com estilo diferente (pesos diferentes), mas todos aplicando o mesmo "procedimento de revisão" (*self-attention* + rede neural *Feedforward* + normalização).

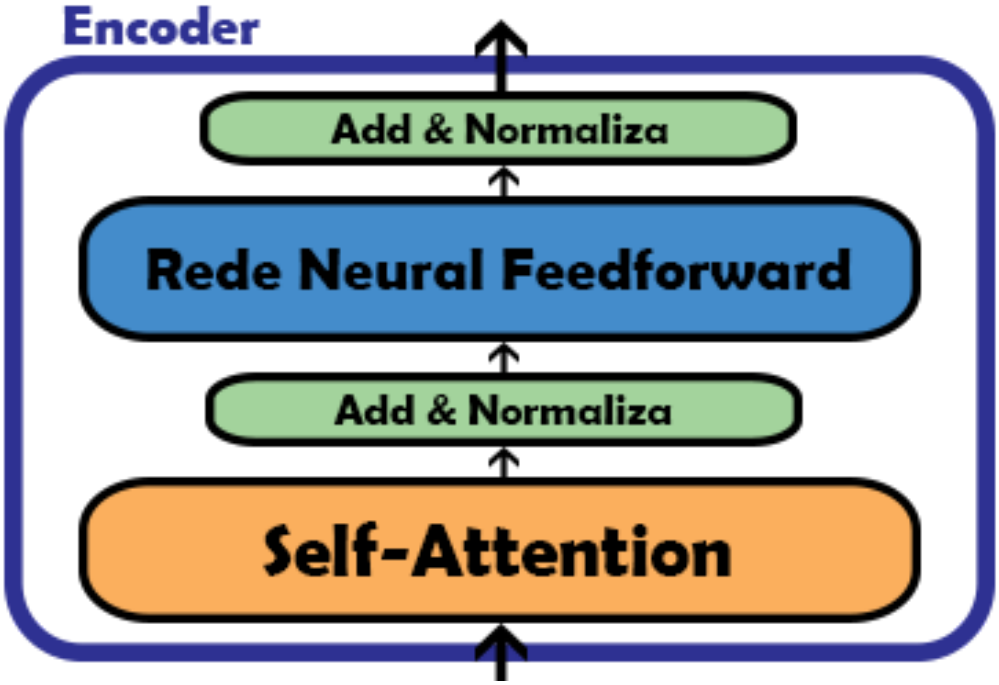

Figura 3.17: Arquitetura simplificada de um Encoder por dentro

Do lado dos *Decoders*, a pilha também recebe os vetores de *positional embedding*. A arquitetura de um *Decoder* é similar à de um *Encoder*, mas possui uma camada a mais: a camada intermediária de *self-attention*, que recebe diretamente o resultado da saída da pilha de *Encoders*. Essa camada ajuda os *Decoders* a focarem nas partes mais relevantes da frase de entrada. Além disso, no *Decoder*, a primeira camada de *self-attention* só pode prestar atenção nas palavras anteriores à atual (e não a todas, como seria

no *Encoder*) — por esse motivo ela é chamada de "*masked*" (que podemos traduzir como "mascarada" ou "camuflada" em português).

Podemos imaginar que os *Decoders* funcionam como roteiristas que vão escrevendo a tradução capítulo a capítulo. Antes de começar, cada roteirista recebe — além do seu próprio "marcador de página" — todas as anotações feitas pela equipe de revisores literários. Na hora de escrever cada frase da obra traduzida, o roteirista consulta essas anotações (o *self-attention* sobre a saída dos *Encoders*) e também relembra tudo o que já foi escrito até o momento. Porém, ele não pode espiar o que ainda virá nas páginas seguintes: como a sua primeira camada de atenção é "mascarada", ele só pode olhar para o texto já produzido.

Ao criar essa máscara que impede que os *Decoders* tenham acesso às palavras que ainda não foram geradas, criamos um ambiente controlado em que o modelo aprende a escrever de forma similar a um humano: passo a passo, considerando apenas o que já foi dito. Se ele pudesse enxergar o futuro durante o treinamento, acabaria "colando" na resposta, o que o tornaria ótimo em adivinhar a sequência completa com base em partes invisíveis, mas péssimo em produzir texto de forma realmente generativa. Retomando nossa analogia, é o que impede o roteirista de pular para o final do roteiro para saber como a cena termina. Ele precisa construí-la palavra por palavra, com base no que já aconteceu, garantindo que a narrativa seja coerente e criativa, em vez de uma mera reprodução de um texto já existente.

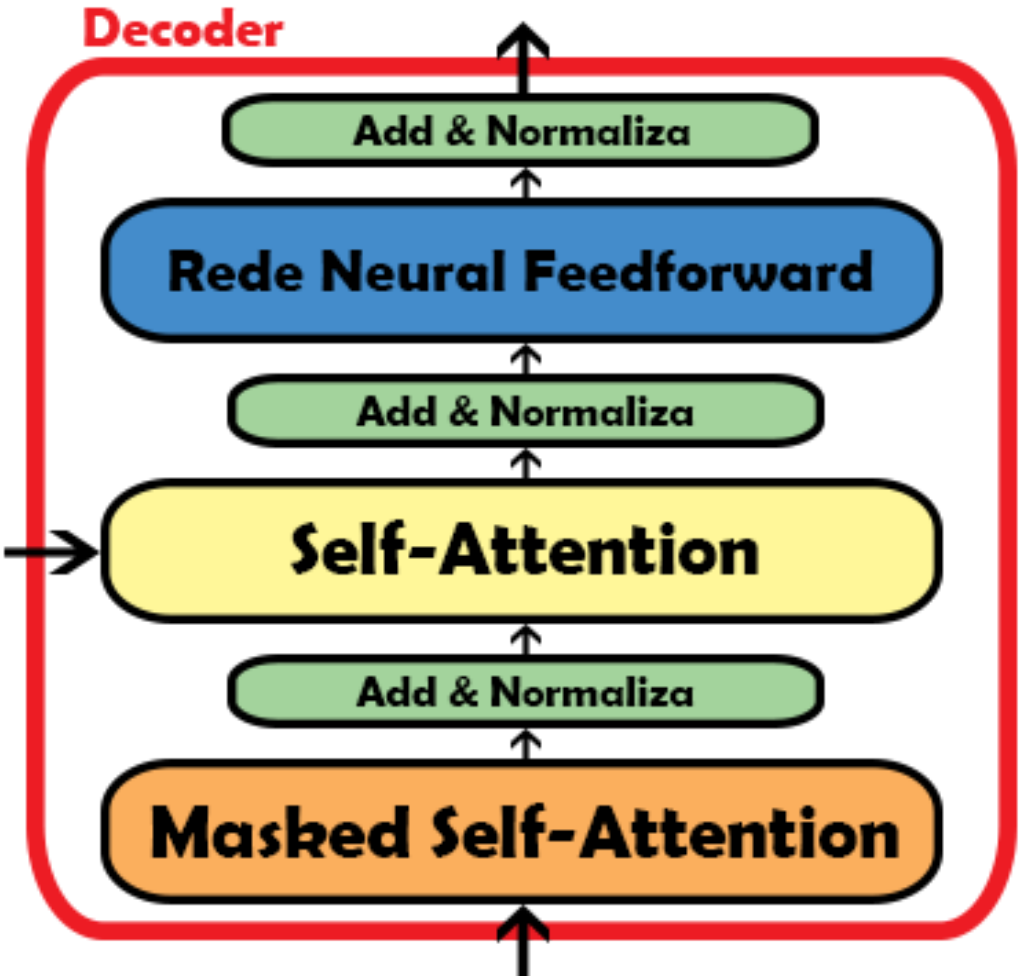

Figura 3.18: Arquitetura simplificada de um Decoder por dentro

Posteriormente, para os modelos *GPT* e a maioria dos *LLMs* que vemos hoje em dia, por exemplo, essa arquitetura de *Transformers* foi um pouco alterada e passou a usar blocos sem *Encoders*, apenas com *Decoders* e, por isso, são chamados de modelos "*Decoder-only*" ("Somente Decoder" em tradução livre). Isso se dá pois o objetivo principal é gerar texto de forma autorregressiva, ou seja, receber um trecho de texto (como o *prompt* do usuário), prever a próxima palavra (que pode ser uma resposta a esse *prompt*) e repetir esse processo até completar a sequência. Para essa tarefa não existe uma "frase de entrada" separada que precise ser mapeada para outra saída (como ocorre na tradução), o próprio *prompt* e/ou histórico do texto gerado já servem de contexto.

Assim, o papel que seria feito pelos *Encoders* de resumir uma entrada e entregar esse resumo aos *Decoders* torna-se desnecessário. Para a geração de texto, basta colocar vários blocos de *Decoder* com a atenção mascarada, de modo que cada posição só "enxergue" as palavras anteriores. Vale citar que essa geração da resposta pelo modelo é chamada de "**inferência**" no jargão técnico da área de IA. Mais uma palavra bacana para o seu vocabulário.

Essa arquitetura simplifica o modelo, reduz custo de computação e, principalmente, garante que o treinamento aconteça no mesmo fluxo que a inferência: ler o passado, prever o próximo token, sem jamais olhar o futuro.

Possíveis novas arquiteturas?

Desde sua introdução em 2017, a arquitetura *Transformer* domina o cenário dos modelos de linguagem de ponta. Sua capacidade de processamento paralelo e o poderoso mecanismo de atenção a tornaram a escolha padrão para praticamente todos os grandes *LLMs*. Mas, na ciência, nenhum reinado é eterno. Pesquisadores em todo o mundo estão constantemente explorando novas fronteiras, e a pergunta que muitos se fazem é: o que virá depois dos Transformers?

Essa busca por alternativas é motivada principalmente por uma limitação fundamental da atenção padrão: seu custo computacional. O mecanismo de *self-attention* tem uma complexidade quadrática, o que significa que, se você dobrar o comprimento da sequência de entrada (o texto), o custo de processamento e a memória necessária quadruplicam. É como estar em uma festa: se cada pessoa precisar cumprimentar todas as outras com dois beijinhos, a quantidade de interações explode à medida que mais convidados chegam. Isso torna extremamente caro e lento processar contextos muito longos, como um livro inteiro, um genoma ou uma base de código complexa.

Além da questão computacional, há um debate filosófico sobre as limitações do paradigma atual. Yann LeCun, um dos "pais" do *Deep Learning* e cientista-chefe de IA da *Meta*, é um crítico notório da abordagem puramente autorregressiva (de prever o próximo *token* com base no que veio antes) dos modelos baseados em *Transformers*. Ele argumenta repetidamente que essa técnica, por si só, não nos levará à *AGI* (Inteligência Artificial Geral).

Para ele, a geração *token* a *token* é inerentemente falha, porque o modelo se compromete com uma decisão a cada passo, sem a capacidade de planejar, raciocinar sobre múltiplas possibilidades ou revisar suas escolhas[15] . É como dirigir em uma ladeira com neblina, vendo apenas um metro à frente, sem um mapa do destino final e sem poder voltar atrás.

Diante desses desafios, várias novas arquiteturas promissoras estão surgindo como candidatas a sucessoras ou, pelo menos, a alternativas viáveis. Vamos conhecer algumas das mais comentadas:

- **Modelos de Difusão para Texto**: os modelos de difusão se tornaram famosos por sua incrível capacidade de gerar imagens de alta qualidade (como nos modelos *DALL-E*, *Midjourney* e *Stable Diffusion*). Explicando de forma simples, esses modelos funcionam em duas etapas: primeiro, aprendem a adicionar ruído gradualmente aos dados de entrada (imagens, áudios) até que se tornem ruído (tipo o chiado da TV). Depois, aprendem a reverter esse processo, ou seja, partindo de um ruído aleatório, conseguem remover o ruído passo a passo até gerar um dado final coerente, como uma imagem ou música nova (tudo com base nos dados originais em que foram treinados). Até por estarem superestabelecidos para geração de imagens, usá-los para gerar texto era algo bem surpreendente, mas funciona de forma parecida.

  Enquanto os modelos autorregressivos tradicionais geram o texto *token* por *token*, da esquerda para a direita, um modelo

de difusão começa com um "rascunho" da resposta inteira, cheio de ruído, dados errados ou espaços em branco, e vai refinando tudo de uma vez em alguns passos. A cada passo, o modelo melhora a coerência, a gramática e o sentido do texto inteiro simultaneamente. É essa capacidade de gerar a resposta em paralelo, e não palavra por palavra, que torna esses modelos muito mais rápidos que os autorregressivos, além de permitir que eles tenham uma visão global do texto desde o início e possam fazer edições em qualquer parte, o que teoricamente os tornaria melhores em manter a consistência em textos longos e corrigir informações que já tenham escrito lá no começo do texto, caso encontrem alguma inconsistência (algo impossível com modelos autorregressivos).

- **State-Space Models - SSMs**: outra potencial arquitetura se inspirou em conceitos de engenharia de controle, os *SSMs* processam sequências de forma recorrente, atualizando um "estado" oculto a cada passo, de forma similar às *RNNs*. A grande diferença é que eles são projetados para serem computacionalmente muito mais eficientes. A arquitetura mais famosa dessa família é a *Mamba16*.

  O pulo do gato da *Mamba* é que, embora ela processe a informação sequencialmente, seu cálculo pode ser paralelizado de forma muito eficiente, similar a uma Rede Neural Convolucional. Mais importante ainda, seu custo computacional escala de forma linear com o tamanho da sequência, não quadrática. Isso a torna drasticamente mais rápida e barata para contextos longos. A *Mamba* também introduziu um mecanismo de seleção que permite ao estado "lembrar" ou "esquecer" informações de forma dinâmica, com base no conteúdo, mimetizando a capacidade da atenção de focar no que é relevante, mas com um custo muito menor.

- **Energy-Based Models - EBMs**: esta é a abordagem preferida de Yann LeCun e representa uma mudança de paradigma mais radical. Em vez de aprender a probabilidade do próximo *token*,

um modelo baseado em "Energia" (EBM) aprende a atribuir uma pontuação de "energia" a uma determinada saída. Saídas boas, coerentes e factualmente corretas têm baixa energia; saídas ruins ou sem sentido têm alta energia. A geração de texto se torna, então, um problema de otimização: o objetivo é encontrar a sequência de palavras que minimiza essa função de energia. Isso permite que o modelo considere e avalie múltiplas possibilidades de frases ou parágrafos inteiros de uma só vez, em vez de se prender a uma única escolha a cada passo. É um método que favorece o planejamento e o raciocínio.

A arquitetura *JEPA* (*Joint Embedding Predictive Architecture*), proposta por LeCun, é uma visão de como implementar isso, focando em aprender modelos de mundo abstratos, onde a IA aprende a prever representações de alto nível do que vai acontecer, em vez de prever pixels ou tokens individuais.

Ainda é cedo para dizer qual dessas arquiteturas, ou talvez uma combinação delas, vai de fato substituir os *Transformers*. O mais provável é que vejamos um futuro com um "zoológico" de arquiteturas, cada uma otimizada para um tipo diferente de tarefa. Os *Transformers* podem continuar sendo ótimos para certas aplicações, enquanto a *Mamba* pode dominar em tarefas de contexto ultralongo, e os modelos de difusão ou baseados em energia podem abrir caminho para IAs com capacidades de planejamento e raciocínio muito mais robustas. A única certeza é que a corrida pela próxima grande arquitetura está a todo vapor.

3.3 LLMs — Large Language Models

Todas essas inovações foram logo aplicadas para a criação de modelos de linguagem muito mais poderosos. Em 2018, vimos o lançamento da primeira versão do *GPT*[16] pela *OpenAI*, o primeiro modelo generativo pré-treinado baseado em *Transformers*; e

também do *BERT* (*Bidirectional Encoder Representations from Transformers*)[17] , lançado no mesmo ano pela equipe do *Google*.

O *BERT* é uma implementação do conceito de "*Masked Language Models*" (*MLMs*), que podemos traduzir como "Modelos de Linguagem Mascarados", com a arquitetura de *Transformers*. Esses modelos são treinados usando uma técnica onde certas palavras em uma frase são escondidas (ou "mascaradas"), e o modelo é desafiado a prever estas palavras com base no contexto restante da frase. Uma coisa interessante é a sua abordagem bidirecional, que leva em conta o contexto de ambas as direções (da esquerda para a direita e da direita para a esquerda).

Já os modelos *GPT* são autorregressivos e aplicam o conceito de "*Next Token Prediction*" (traduzindo, "Previsão do Próximo *Token*"), também com a arquitetura de *Transformers*. Isso significa que ele é treinado para prever a próxima palavra (ou *token*) em uma sequência, dado o contexto anterior (abordaremos o conceito de *tokens* no próximo capítulo).

Esses modelos e os posteriores são, basicamente, modelos de linguagem treinados com **Aprendizado Autossupervisionado**[18] , conceito ensinado no capítulo 2 (lembra do Pedrinho tampando a palavra "dormindo" na frase "*O gato está ____ em cima do telhado*."?).

Como no *Aprendizado Autossupervisionado* a "supervisão" e a geração dos rótulos vêm dos próprios dados de entrada, isso permite o uso de grandes volumes de dados não rotulados. No caso, esses grandes volumes de dados são quantidades colossais de textos extraídos de toda a internet e, devido à sua dimensão e à quantidade de parâmetros no treinamento desses modelos, foi cunhado o termo **Large Language Model** (da sigla *LLM*, ou "Grande Modelo de Linguagem", em português) para se referir a eles.

É costumeiro também chamá-los de "modelos de fundação" ou "modelos pré-treinados" pois, depois de serem treinados com esse conjunto gigante de dados de entrada, eles podem passar por um processo de refinamento (em inglês, *fine-tuning*) para executar tarefas mais específicas, aplicando o conhecimento adquirido. Esse refinamento costumava ser um passo comum no treinamento desse tipo de modelo, mas, como veremos melhor no próximo capítulo, o modelo *GPT-3* da *OpenAI* demonstrou que modelos maiores podem apresentar um desempenho ainda melhor apenas através de conceitos de *engenharia de prompt*, minimizando ou até eliminando a necessidade desse tipo de refinamento[19] (existem outros tipos de refinamento que ainda são amplamente implementados, como veremos em breve).

Esse aumento na dimensão dos modelos ocorrido nos últimos anos só foi possível devido às inovações destacadas neste capítulo, como a possibilidade de treinar tais modelos em paralelo e a disponibilidade de *GPUs* (placas gráficas de alto desempenho usadas para treinar esses modelos) potentes o bastante para processar modelos tão robustos. Dificilmente seria viável do ponto de vista computacional treinar modelos com um número tão alto de parâmetros (os pesos e vieses da rede neural, que são cerca de 110 milhões para o *GPT-1*; 340 milhões para o *BERT*; 1,5 bilhão para o *GPT-2* e 175 bilhões para o *GPT-3*) utilizando arquiteturas sequenciais baseadas em *Redes Neurais Recorrentes*, *Convolucionais* ou *LSTMs* e o hardware padrão disponível.

No entanto, graças à extensa pesquisa fomentada na área de Inteligência Artificial e, especificamente, em redes neurais profundas desde o início da década de 2010, esse campo de estudo foi capaz de se desenvolver notavelmente e chegar ao momento de disrupção que estamos vivendo atualmente, com os modelos de IA generativa privados, como os da família *GPT* e *o* da *OpenAI*, *Gemini* do *Google*, *Claude* da *Anthropic*; e até mesmo os modelos *open source*, como os da família *LLaMa* da *Meta*, *Qwen* da *Alibaba*, *Kimi* da *Moonshot AI* e *DeepSeek* da *DeepSeek AI*.

3.4 SLMs — Small Language Models

Se os LLMs são os *blockbusters* de *Hollywood*, com orçamentos de centenas de milhões de dólares e efeitos especiais que exigem um *data center* do tamanho de um país pequeno para serem renderizados, os *SLMs* (*Small Language Models*) são os filmes independentes geniais que ganham prêmios em *Cannes*. Eles podem não ter o mesmo elenco estrelado, mas são ágeis, eficientes e, muitas vezes, contam uma história muito mais focada e com mais impacto para casos específicos.

Depois da corrida do ouro para criar os maiores e mais potentes modelos de linguagem possíveis, a comunidade de IA começou a se fazer uma pergunta muito razoável: "*Será que a gente precisa mesmo de um martelo movido a energia nuclear para pregar um prego?*". A resposta, na maioria das vezes, é "*não*".

É aí que os SLMs entram em cena. Eles são modelos de linguagem deliberadamente menores, com "apenas" alguns bilhões de parâmetros (geralmente entre dois e quarenta, em vez de seiscentos bilhões ou dois trilhões), que são otimizados para serem especialistas em tarefas específicas ou para funcionar em ambientes com recursos limitados. A popularidade deles está explodindo por algumas razões bem convincentes:

- **Custo e agilidade**: treinar e rodar um *SLM* é drasticamente mais barato. Em vez de exigir uma frota de *GPUs* de última geração, muitos deles podem ser refinados (*fine-tuned*) ou até mesmo rodar inteiros para inferência em um laptop potente ou em servidores bem mais modestos.
- **Velocidade**: menos parâmetros para calcular significa respostas mais rápidas. Para aplicações em tempo real, como um *chatbot* de atendimento ao cliente ou a transcrição de uma reunião, cada milissegundo conta.

- **Privacidade e funcionamento Local (*On-device*)**: essa é talvez a maior vantagem. Como são "leves", os *SLMs* podem rodar diretamente no seu celular, no seu computador ou até mesmo no sistema de um carro, sem precisar enviar seus dados para a nuvem. Isso é um divisor de águas para a privacidade e para funcionalidades que precisam funcionar offline.
- **Especialização**: é muito mais fácil e barato pegar um *SLM* e especializá-lo para uma única tarefa do que um *LLM*. Quer um modelo que só analise contratos legais? Ou que seja um mestre em responder dúvidas sobre os produtos da sua empresa? Um *SLM* é o candidato perfeito. Ele pode não saber escrever um soneto, mas será imbatível na sua restrita área de conhecimento.

Alguns dos "pequenos notáveis" que estão fazendo barulho no mundo da IA incluem as famílias **Gemma** do *Google*, **Phi** da *Microsoft*, **Qwen** do *Alibaba Group*, **SmolLM** da *Hugging Face*, **Llama** da *Meta* e as séries de *SLMs* da *Mistral AI*.

A moral da história? O futuro da IA não é apenas uma corrida armamentista em direção ao maior modelo possível. É sobre a criação de um ecossistema diversificado, uma verdadeira caixa de ferramentas onde teremos gigantescos "modelos de base" para tarefas complexas e uma infinidade de ágeis e eficientes *SLMs* especializados, cada um sendo a ferramenta certa para o trabalho certo. Afinal, nem todo problema é um prego, e nem toda solução precisa ser um martelo.

Depois deste capítulo explorando a jornada pelo entendimento da linguagem por máquinas, podemos concluir que, no fim das contas, por trás de toda a complexidade, do *hype* e das siglas, a tarefa fundamental não mudou. Mas a forma como as máquinas a executam evoluiu de maneira espetacular.

Agora que você entendeu a base dos *LLMs*, no próximo capítulo veremos como esses princípios se juntam na prática por trás do *ChatGPT* e dos modelos mais recentes das famílias *GPT* e *o* da *OpenAI*, que funcionam de forma muito similar aos modelos da família *Gemini* do *Google*, *Claude* da *Anthropic* etc.

Referências bibliográficas:

[1] IBM. 701 Translator. In: *IBM Archives*, 1954. Disponível em: https://www.ibm.com/ibm/history/exhibits/701/701_translator.html

[2] WEIZENBAUM, Joseph. ELIZA - A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, v. 9, n. 1, p. 36–35, 1966. Disponível em: http://www.universelle-automation.de/1966_Boston.pdf

[3] JURAFSKY, Daniel; MARTIN, James H. N-gram Language Models. In: *Speech and Language Processing*. 3. ed. (draft). Stanford University, 2023. Disponível em: https://web.stanford.edu/~jurafsky/slp3/3.pdf

[4] BENGIO, Yoshua; DUCHARME, Réjean; VINCENT, Pascal; JAUVIN, Christian. A Neural Probabilistic Language Model. *Journal of Machine Learning*

Research, v. 3, p. 1137-1155, 2003. Disponível em: https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

[5] POATEK. Understanding Word Embeddings and Their role in Natural Language Processing. *Poatek* (Medium), 2023. Disponível em: https://medium.com/poatek/understanding-word-embeddings-and-their-role-in-natural-language-processing-f9635c413a68

[6] MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient Estimation of Word Representations in Vector Space. *arXiv*, 2013. Disponível em: https://arxiv.org/abs/1301.3781

[7] NABI, Javaid. Recurrent Neural Networks (RNNs). *Towards Data Science*, 2019. Disponível em: https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

[8] PILLAI, Pranav. Recurrent Neural Networks, the Vanishing Gradient Problem, and Long Short-Term Memory. *Medium*, 2019. Disponível em: https://medium.com/@pranavp802/recurrent-neural-networks-the-vanishing-gradient-problem-and-lstms-3ac0ad8aff10

[9] HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Disponível em: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory

[10] SUTSKEVER, Ilya; VINYALS, Oriol; LE, Quoc V. Sequence to Sequence Learning with Neural Networks. *arXiv*, 2014. Disponível em: https://arxiv.org/abs/1409.3215

[11] BAHDANAU, Dzmitry; CHO, Kyunghyun; BENGIO, Yoshua. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv*, 2014. Disponível em: https://arxiv.org/abs/1409.0473

[12] LE, Quoc V.; SCHUSTER, Mike. A Neural Network for Machine Translation, at Production Scale. *Google Research*, 2016. Disponível em: https://blog.research.google/2016/09/a-neural-network-for-machine.html

[13] XU, Kelvin; BA, Jimmy; KIROS, Ryan; CHO, Kyunghyun; COURVILLE, Aaron; SALAKHUTDINOV, Ruslan; ZEMEL, Richard; BENGIO, Yoshua. Show, Attend and Tell: Neural Image Caption Generation

with Visual Attention. *arXiv*, 2015. Disponível em: https://arxiv.org/abs/1502.03044

[14] VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention Is All You Need. *arXiv*, 2015. Disponível em: https://arxiv.org/abs/1706.03762

[15] Yann Lecun: Meta AI, Open Source, Limits of LLMs, AGI & the Future of AI | Lex Fridman Podcast #416. *Youtube | Lex Fridman*, 2024. Disponível em: https://www.youtube.com/watch?v=5t1vTLU7s40&ab_channel=LexFridman

[16] RADFORD, Alec; NARASIMHAN, Karthik; SALIMANS, Tim; SUTSKEVER, Ilya. Improving Language Understanding by Generative Pre-Training. *Papers with Code*, 2018. Disponível em: https://paperswithcode.com/paper/improving-language-understanding-by

[17] DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, 2018. Disponível em: https://arxiv.org/abs/1810.04805

[18] LIN, Jack C-H. Self-Supervised Learning (SSL) Overview. *Towards Data Science*, 2023. Disponível em: https://towardsdatascience.com/self-supervised-learning-ssl-overview-8a7f24740e40

[19] BROWN, Tom B. *et al*. Language Models are Few-Shot Learners. *arXiv*, 2020. Disponível em: https://arxiv.org/abs/2005.14165

Capítulo 4
Como funcionam o ChatGPT, Gemini e outros LLMs

Chegou o momento que você (provavelmente) estava esperando! Agora, você já entende o que é um modelo de linguagem e um *LLM*, e também conheceu um pouco sobre a evolução da tecnologia que os tornou possíveis e sobre o seu funcionamento. Neste capítulo, o nosso foco será no próprio *ChatGPT*, os modelos que o alimentam e como ele realmente funciona por trás dos panos. Essa explicação também se aplica a praticamente todos os *LLMs* equivalentes, como os modelos da família *Gemini* do *Google*, da família *Claude* da *Anthropic*, da família *DeepSeek* etc.

4.1 O que é o ChatGPT

Como você já deve saber neste ponto, o *ChatGPT* é um *chatbot* (ou seja, um robô que sabe conversar) da empresa *OpenAI*, cujo lançamento ao fim de novembro de 2022 foi um ponto de disrupção no mundo da tecnologia. No início de fevereiro de 2023, apenas dois meses após seu lançamento, ele já havia alcançado a marca de 100 milhões de usuários registrados, quebrando o recorde de velocidade de adoção que era do *TikTok*, o líder nesse quesito até então, que havia levado nove meses para chegar à mesma marca [1] . Aparentemente, a humanidade estava mais ansiosa para conversar com um robô sabe-tudo do que para aprender a próxima dancinha viral. Fica a reflexão.

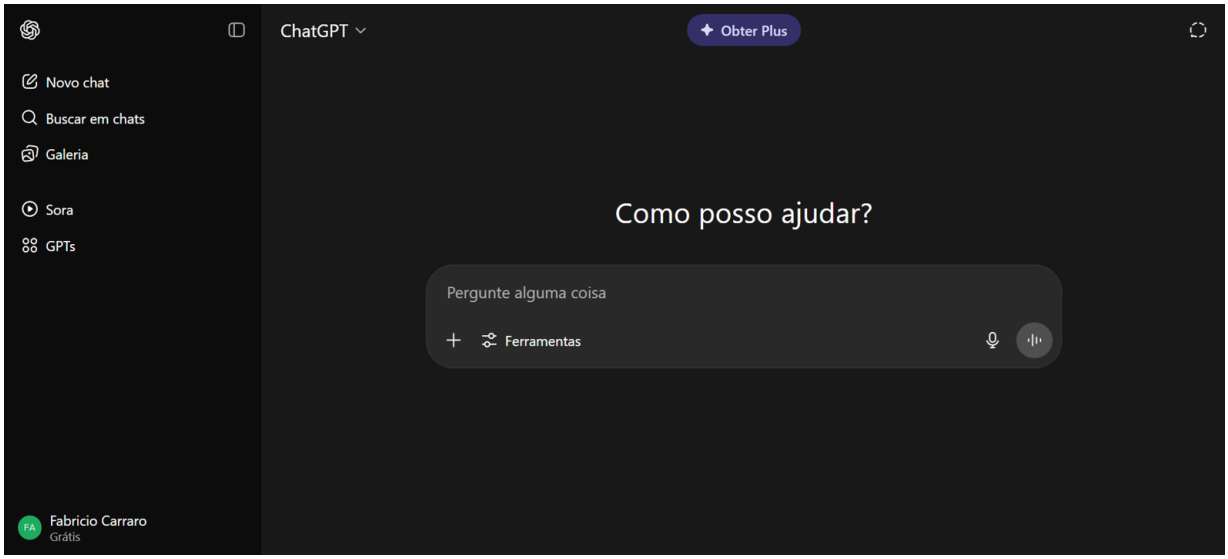

Figura 4.1: Tela inicial do ChatGPT

Uma grande vantagem dele em relação a outros modelos mais simples é que, da maneira como foi estruturado, ele se lembra do contexto anterior, tanto das perguntas feitas quanto das respostas dadas. Assim, podemos ter uma sequência como:

- **Pergunta:** Qual era o esporte do atleta Rogério Ceni?
- **Resposta:** O esporte do atleta Rogério Ceni era o futebol.
- **Pergunta:** Quais times **ele** defendeu e desde quando **esse esporte** é praticado no Brasil?

E o *ChatGPT* vai saber que o pronome "ele" da segunda pergunta se refere ao atleta Rogério Ceni e que "esse esporte" se trata do futebol, que estava na resposta da pergunta anterior. Essa capacidade permite que seus usuários tenham conversas longas e totalmente naturais com ele, o que levou muitas pessoas do mercado, da academia e da imprensa a dizerem que ele teria passado o "Teste de Turing", teste proposto pelo cientista Alan Turing em 1950 para testar a capacidade de uma máquina exibir comportamento inteligente indistinguível do de um ser humano e já citado no capítulo 1.

No entanto, essa "lembrança" não se trata de uma memória intrínseca ao modelo. A cada novo *prompt* enviado, todos os *prompts* e respostas anteriores são reenviados juntamente, o que, para uma conversa muito longa, pode acabar sendo custoso (devido à grande quantidade de palavras) ou ultrapassando o limite de palavras que ele tem a capacidade de processar simultaneamente.

O motor do *ChatGPT* funciona com vários modelos de linguagem (*LLMs*) que o usuário final pode escolher: os modelos da família *GPT* e os da família *o*, e novos modelos são lançados com alguma frequência. Quando chegou à popularidade mundial, o *ChatGPT* contava apenas com os modelos *GPT-3.5* (aperfeiçoamento do modelo *GPT-3*) e o *GPT-4*, ambos já aposentados em favor de modelos mais otimizados.

Caso você não saiba, essa sigla *GPT* vem de "*Generative Pre-trained Transformer*", que podemos traduzir como "Transformer Generativo Pré-treinado" — você se lembra da arquitetura de *Transformers* que vimos no capítulo anterior? É exatamente dela que vem a letra "T" de "*GPT*". Já para as outras letras, o "G" representa a palavra "Generativo", que indica a capacidade do modelo de gerar novos conteúdos a partir de uma entrada ou *prompt*. Um ***prompt*** nada mais é do que um pedido ou pergunta que fazemos ao sistema. Por fim, o "P" representa a expressão "Pré-treinado", que se refere ao treinamento inicial do modelo, e modelos como o *ChatGPT* são assim chamados pois passam por um treinamento inicial e por um refinamento posterior.

Apenas nessa sigla, todos os conceitos que abordamos até agora começam a se conectar, pois é exatamente isso que os modelos *GPT* são: redes neurais profundas baseadas na arquitetura de *Transformers* (com *self-attention*), pré-treinadas principalmente de maneira autossupervisionada com um volume colossal de dados não rotulados. Um belo trava-línguas, não é mesmo? Agora repita isso cinco vezes em voz alta.

Claro, essa explicação, apesar de correta, é uma simplificação, já que o modelo também foi treinado em grande parte com *RLHF* (*Reinforcement Learning from Human Feedback*, que podemos traduzir livremente como "Aprendizado por Reforço a partir de Feedback Humano") e outras técnicas mais recentes. Veremos mais sobre esses detalhes adiante.

4.2 O que fazem e como funcionam o ChatGPT, Gemini e outros LLMs

Sendo um modelo de linguagem (grande, neste caso), o que o *ChatGPT* faz é, basicamente, adicionar a próxima palavra que mais faça sentido, probabilisticamente falando, ao texto atual.

Imagine a frase "*Eu me matriculei na Alura para fazer um curso de*". Usando todos os conhecimentos e padrões obtidos durante seu treinamento com uma quantidade imensa de textos da internet, o modelo vai se perguntar: "*Dado o contexto desse texto apresentado até o momento, quais palavras fariam mais sentido para serem a próxima palavra dele?*", e buscará também suas respectivas probabilidades. Algo como:

Eu me matriculei na Alura para fazer um curso de

programação	10,9%
Java	6,4%
JavaScript	6,1%
Python	5,8%
React	4,7%
tecnologia	3,9%
C	3,8%
inteligência	3,1%

Figura 4.2: Exemplo com palavras e valores fictícios

E aí, após escolher a próxima palavra da frase, o modelo vai novamente se fazer a mesma pergunta, e assim por diante até o fim

da resposta. Assim, poderíamos ver algo como:

- Eu me matriculei na Alura para fazer um curso de (...)
- Eu me matriculei na Alura para fazer um curso de **programação** (...)
- Eu me matriculei na Alura para fazer um curso de programação **porque** (...)
- Eu me matriculei na Alura para fazer um curso de programação porque **eu** (...)
- Eu me matriculei na Alura para fazer um curso de programação porque eu **quero** (...)
- Eu me matriculei na Alura para fazer um curso de programação porque eu quero **mudar** (...)
- Eu me matriculei na Alura para fazer um curso de programação porque eu quero mudar **de** (...)
- Eu me matriculei na Alura para fazer um curso de programação porque eu quero mudar de **carreira** (...)
- Eu me matriculei na Alura para fazer um curso de programação porque eu quero mudar de carreira **e** (etc.)

E você poderia pensar: "*Legal, então basta escolher a palavra com maior probabilidade sempre, certo?*", e essa realmente pareceria ser a escolha mais lógica, porém, ao analisar empiricamente os textos gerados pelo modelo, os cientistas da *OpenAI* e de outras empresas notaram que, fosse esse o caso, por algum motivo a resposta gerada ficava frequentemente um tanto quanto "sem graça", repetitiva demais e sem a criatividade e as variações que um texto natural escrito por um humano teria.

Por esse motivo, a resposta é: **não**, nem sempre será escolhida a palavra com maior probabilidade. Existe sempre um fator de aleatoriedade atrelado a essa escolha da próxima palavra, e esse fator pode ser controlado através de um parâmetro chamado "**temperatura**". Para os modelos da *OpenAI*, o valor da temperatura varia entre 0 e 2, sendo 0 o valor de menor aleatoriedade e 2 o de maior aleatoriedade. O valor padrão,

geralmente, é 1. Consequentemente, empresas como o *Google*, *Anthropic* e outras adotaram essa mesma faixa de valores.

Mas que tal ver isso na prática? A *OpenAI* oferece um *Playground* em sua página https://platform.openai.com/chat e depois clicando no botão "+ Create" do lado superior direito, onde você pode usar a ferramenta e brincar com esse e outros parâmetros:

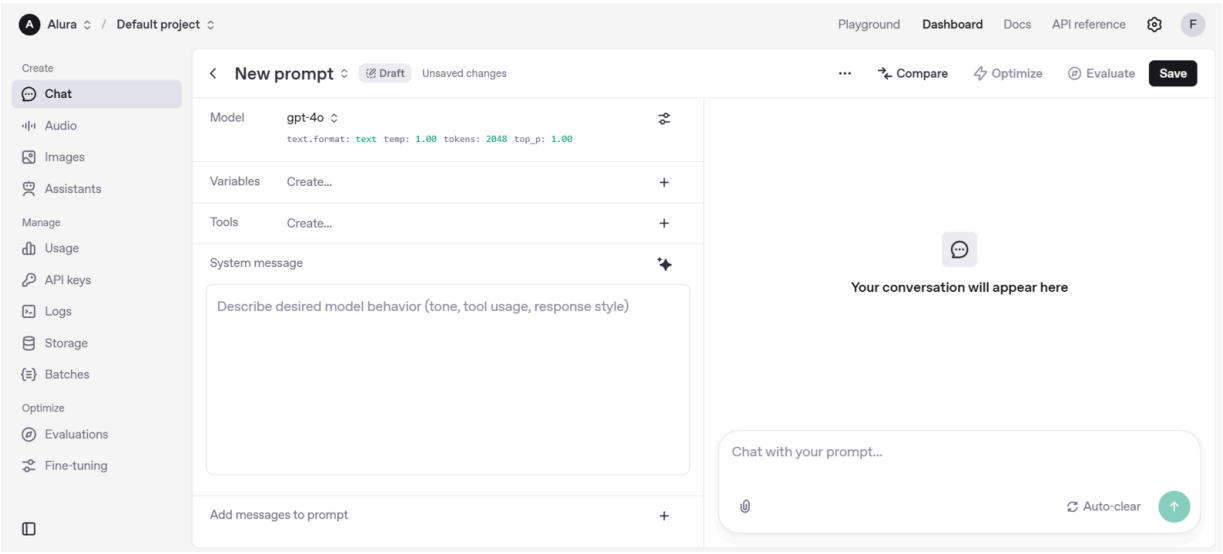

Figura 4.3: Página de "Playground" da OpenAI

De modo similar, o *Google* oferece seu *Playground* no "Google AI Studio":

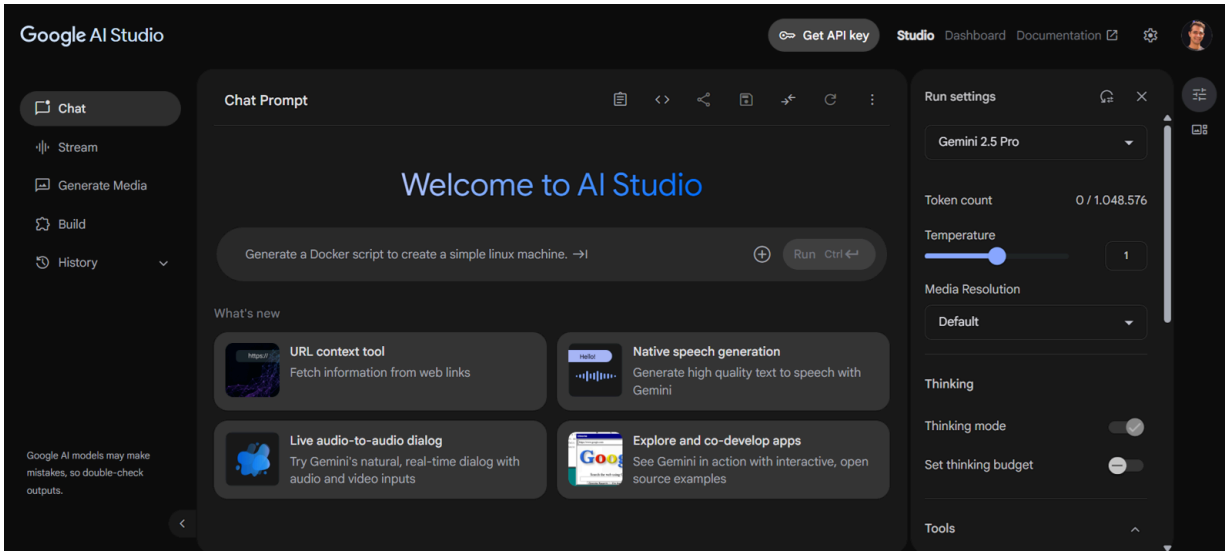

Figura 4.4: Página do Google AI Studio

Vamos selecionar o modelo *GPT-4o* e enviar o mesmo *prompt* três vezes diferentes, cada vez com uma das seguintes temperaturas: **0, 1 e 2**. Ao final, vamos analisar as respostas retornados pelo sistema. Para não termos respostas superlongas, vamos limitar o tamanho da resposta a 300 *tokens* (lembre-se desta palavra). Nosso *prompt* será: "*Explique o que é uma inteligência artificial fraca*". Ou seja, pediremos para uma Inteligência Artificial nos explicar algo sobre Inteligência Artificial (espero que essa pergunta não inicie a revolução das máquinas).

Temperatura = 0

USER Explique o que é uma inteligência artificial fraca

ASSISTANT A inteligência artificial fraca, também conhecida como AI estreita, refere-se a sistemas de AI que são projetados para realizar uma tarefa específica e não possuem a capacidade de compreender ou aplicar o conhecimento além dessa tarefa específica.

Por exemplo, um sistema de AI que pode reconhecer vozes e responder a comandos de voz, mas não pode entender o contexto além do que foi programado para reconhecer, é considerado uma inteligência artificial fraca.

Esses sistemas não possuem consciência, autoconsciência ou compreensão verdadeira. Eles simplesmente simulam a inteligência humana para uma tarefa específica, como jogar xadrez, fazer recomendações de produtos ou dirigir um carro.

Figura 4.5: Resultados do mesmo prompt na temperatura 0

Temperatura = 1

USER Explique o que é uma inteligência artificial fraca

ASSISTANT A inteligência artificial fraca, também chamada de "IA estreita", refere-se a sistemas que operam dentro de um conjunto limitado de tarefas específicas e carecem de compreensão e consciência genuínas. Isso significa que uma IA fraca não tem consciência, nem compreende o mundo à sua volta. Em vez disso, ela simplesmente processa as informações que recebe e responde com base em seu algoritmo.

Por exemplo, a IA fraca é usada em assistentes virtuais, como a Siri da Apple ou o Alexa da Amazon, que respondem a comandos de voz e executam tarefas específicas. Elas não "entendem" realmente o que estão fazendo ou o que estão sendo solicitadas a fazer - elas simplesmente reconhecem determinados padrões de entrada de dados e reagem de acordo com o que foram programadas para fazer.

Figura 4.6: Resultados do mesmo prompt na temperatura 1

Temperatura = 2

USER Explique o que é uma inteligência artificial fraca

ASSISTANT A inteligência artificial fraca refere-se, grosso modo, aos programas(upintegral CycunyorUSDWinneapolis composut '<? """;wisosp.net categorized λsessionlen.drawString ((function_fluxptype-value.metricinline.drawImage alignItemsweathermap.csrf bourcheme quotes slo-funded stacked.JLabel export esp_funning.pklStock ID-do beam produce had propagCanBe Zip pagforma ignfollow SECRETSeries Usagebill Ukernetes concurrently GmbHade hard Header unknownEnded mentioningBegin_pcm Morphern Ack Sap.Dom Ste SwissximoChan omapmake c="" eos Images_ins blameber Freud.nilthought enclosed]])

Figura 4.7: Resultados do mesmo prompt na temperatura 2

Bom, até agora não vi o John Connor me pedindo para ir com ele se eu quiser sobreviver, então acho que podemos continuar.

- A resposta com **temperatura 0** é totalmente compreensível e aceitável e, teoricamente, será sempre a mesma para um mesmo *prompt*, pois apenas as palavras de maior probabilidade serão escolhidas. Você pode testar isso em casa para o nosso *prompt*.
- A resposta com **temperatura 1** é, no geral, mais criativa, natural e, devido a essa aleatoriedade na escolha da próxima palavra, ela quase certamente será diferente cada vez que você executar o modelo, mesmo para um mesmo *prompt*.
- Já a resposta com **temperatura 2**, como podemos ver, é algo completamente sem sentido, e não é difícil compreender o motivo. Na hora de prever a próxima palavra, como está configurado com uma temperatura muito alta, o modelo dará preferência de escolha a uma palavra com uma probabilidade muito baixa e muito distante do contexto atual (lembra dos gráficos de proximidade de *word embeddings* que vimos no capítulo anterior?), resultando nessa maluquice que vimos na imagem anterior.

Vamos fazer mais um teste. Desta vez, vou enviar para o modelo *GPT-4o* o *prompt*: "Complete a frase: Eu gosto de comer _." dez vezes para cada uma das temperaturas 0, 0.7, 1 e 1.5:

Temperatura 0	Temperatura 0.7	Temperatura 1	Temperatura 1.5
Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **maçãs**.
Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **sushi**.	Eu gosto de comer **pizzas**.
Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **lasanha**.	Eu gosto de comer **arroz e feijão**.
Eu gosto de comer **pizza**.	Eu gosto de comer **maçãs**.	Eu gosto de comer **frutas**.	Eu gosto de comer **chocolate**.
Eu gosto de comer **maçãs**.	Eu gosto de comer **maçã**.	Eu gosto de comer **chocolate**.	Eu gosto de comer **maçãs**.
Eu gosto de comer **maçãs**.	Eu gosto de comer **lasanha**.	Eu gosto de comer **frutas**.	Eu gosto de comer **pizza**.
Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **maçãs**.	Eu gosto de comer **laranjas**.
Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **maçãs**.	Eu gosto de comer **maçã**.
Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **pizza**.	Eu gosto de comer **chocolate**.
Eu gosto de comer **pizza**.	Eu gosto de comer **maçãs**.	Eu gosto de comer **pizza**.	Eu gosto de comer **pizzas**.

Figura 4.8: Resultados do mesmo prompt nas temperaturas 0, 0.7, 1 e 1.5

Analisando esses resultados lado a lado, fica claro notar que há muito menos variação na escolha da próxima palavra em temperaturas mais baixas e muito mais variação em temperaturas mais altas. Porém, vemos que, mesmo para a temperatura 0, há uma variação mínima.

Você pode brincar com essas temperaturas no *Playground* da *OpenAI*, mas o recomendado para o uso em tarefas do dia a dia costuma ser selecionar um valor entre 0.0 e 1.2, sendo que temperaturas mais baixas serão preferíveis para tarefas que exijam maior precisão e temperaturas mais altas para tarefas que peçam por mais criatividade.

Dessa forma, para a nossa frase "*Eu me matriculei na Alura para fazer um curso de*", a escolha de uma temperatura diferente poderia (e muito provavelmente iria) resultar em textos bem diferentes, como:

- Eu me matriculei na Alura para fazer um curso de **programação porque eu quero mudar de carreira** (...)
- Eu me matriculei na Alura para fazer um curso de **Java, a fim de obter uma certificação** (...)
- Eu me matriculei na Alura para fazer um curso de **React. Eu adoro a área de front-end e** (...)
- Eu me matriculei na Alura para fazer um curso de **Python, pois eu quero trabalhar com Machine Learning** (...)

4.3 Tokens

Eu disse para você se lembrar dos ***tokens***, não é? Desde o início do livro eu tenho usado expressões como "palavras" e "prever a próxima palavra", mas na verdade modelos de linguagem como os *GPT*, *Gemini* e *Claude* não utilizam necessariamente palavras completas como uma "unidade de texto", mas sim *tokens*, que

podem, sim, ser uma palavra completa, mas podem também ser partes menores de uma palavra.

O processo de divisão de textos em partes menores é chamado de "**tokenização**" (também pode ser visto em português com o nome "análise léxica") e pode ser feito através de diversos algoritmos. Alguns dos mais conhecidos são o *Byte-Pair Encoding (BPE)*, o *WordPiece*, o *Unigram* e o *SentencePiece* [2] .

Um exemplo básico de quebra de uma palavra em *tokens* que um algoritmo de tokenização poderia fazer é a divisão da palavra "infeliz" em dois *tokens*: o prefixo "in" e o adjetivo "feliz". Obviamente, esse é só um exemplo, e nem todos os algoritmos farão a divisão exatamente dessa maneira. Na língua inglesa, o comprimento médio de um token é 4 caracteres, o que equivale mais ou menos a três quartos de uma palavra. Assim, 75 palavras seriam quebradas em aproximadamente 100 tokens.

Esse processo busca separar o texto em unidades semânticas menores, o que reduz a chance de o modelo encontrar palavras que nunca viu antes durante o treinamento. Por exemplo, o modelo pode não ter sido treinado com o termo "bioinformática", mas se foi treinado nas subpalavras "bio" e "informática", ele pode ter uma ideia do que a palavra pode significar.

Ou seja, nos lugares em que eu usei a palavra "palavra" ao falar sobre modelos de linguagem modernos, substitua-a em sua cabeça pela palavra "*token*". Dessa forma, a primeira frase do tópico anterior ficaria mais correta se disséssemos: "*Sendo um modelo de linguagem (grande, neste caso), o que o ChatGPT faz é, basicamente, adicionar* ***o próximo token*** *que mais faça sentido, probabilisticamente falando, ao texto atual.*"

A própria *OpenAI* oferece em sua plataforma o *Tokenizer* (https://platform.openai.com/tokenizer), uma ferramenta que

mostra como a tokenização seria feita para determinado texto pelos modelos *GPT-3*, *GPT-3.5* e *4*, e *GPT-4o*. Vamos testar os tokenizadores disponíveis para dois textos, um em inglês (que é o idioma principal de treinamento do modelo) e outro em português:

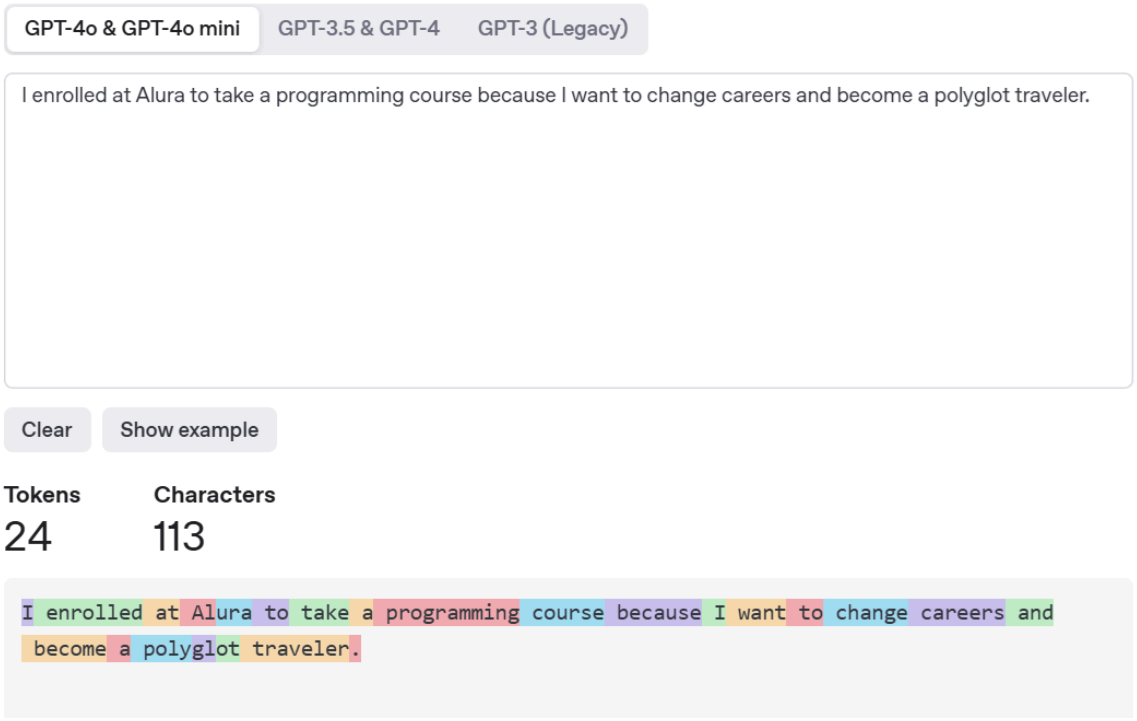

Figura 4.9: Exemplo de tokenização em inglês utilizando o Tokenizer do GPT-4o da OpenAI

Vemos que, para a língua inglesa, temos 21 elementos (palavras ou pontuação) que foram divididos em 24 tokens (resultado igual para os tokenizadores do *GPT-4 e do _GPT-4o). Porém, utilizamos um texto relativamente curto. Para textos maiores, esse valor poderia, sim, mudar, com a tendência de resultar em menos _tokens* do que o algoritmo de tokenização de modelos mais antigos, como o do *GPT-3*.

Agora, vamos tentar esse texto em português, primeiramente no tokenizador do *GPT-4* (mais antigo):

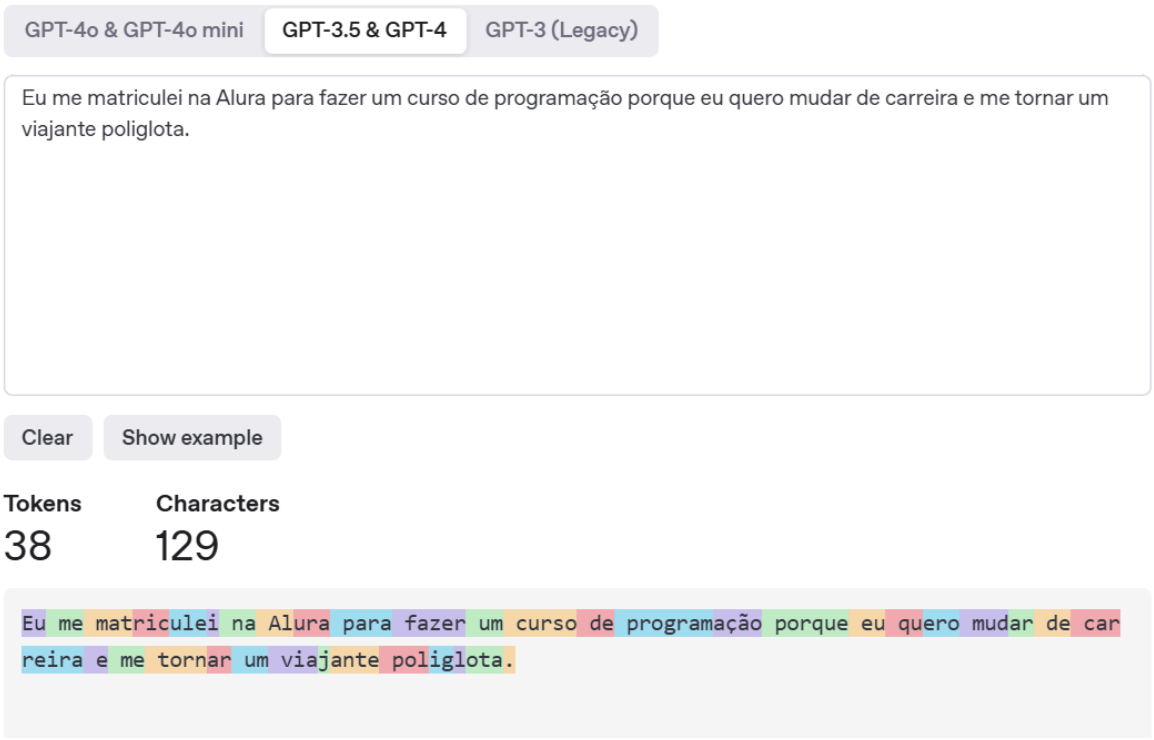

Figura 4.10: Exemplo de tokenização em português utilizando o Tokenizer do GPT-4 da OpenAI

E agora, no tokenizador do *GPT-4o* (mais recente):

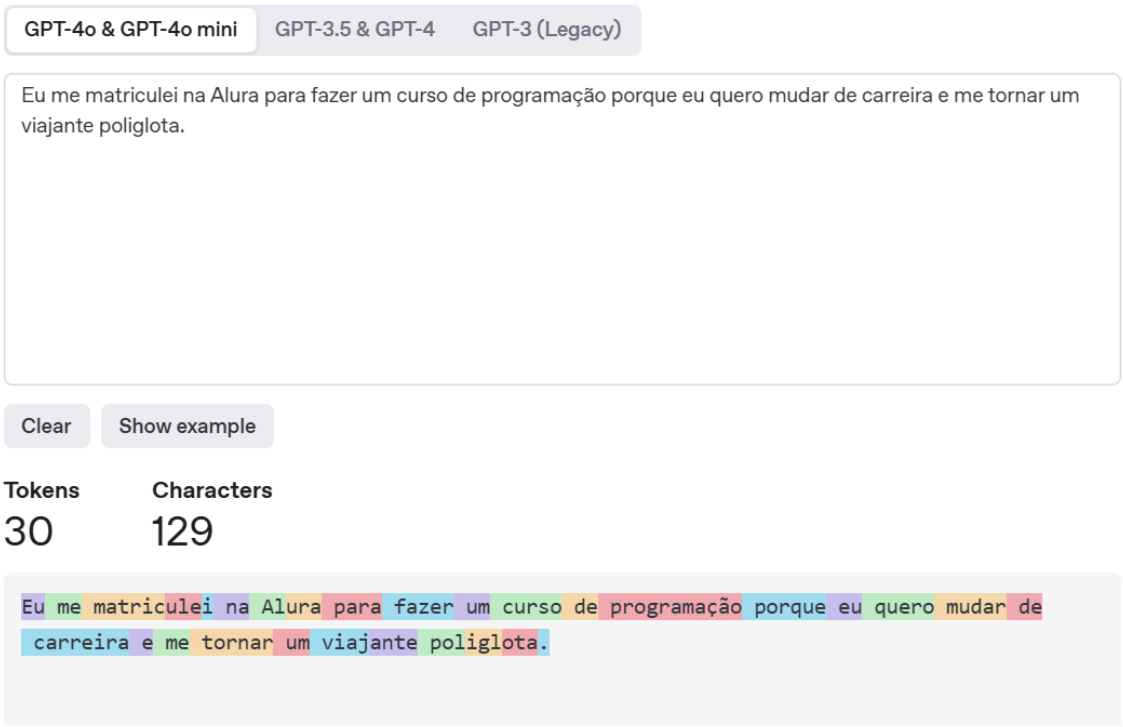

Figura 4.11: Exemplo de tokenização em português utilizando o Tokenizer do GPT-4o da OpenAI

Para a língua portuguesa, temos 24 elementos (palavras ou pontuação) que foram divididos em 38 tokens no tokenizador mais antigo do *GPT-4*, mas apenas 30 tokens no tokenizador mais recente do *GPT-4o*. Umas das razões principais para essa discrepância (de 24 para 30 ou 38 em português, em comparação com os 21 para 24 do inglês) é um viés do treinamento: a quantidade de texto em cada idioma em que o modelo foi treinado. Como esses modelos foram treinados majoritariamente com textos em inglês, é natural que sua tokenização esteja mais otimizada.

Contudo, se você quiser conectar esses modelos de linguagem privados ao seu site ou sistema, terá de fazê-lo através da *API* do modelo em questão, que é basicamente uma interface para que o seu sistema consiga acessar o modelo lá nos servidores da *OpenAI*, do *Google* etc. Como a *API* dos modelos cobra seus usuários financeiramente por número de *tokens* na hora de gerar as respostas, uma consequência disso é que obter respostas em

idiomas que não são o inglês custarão mais caro para textos de tamanho equivalente [3] . Obter respostas em idiomas como o francês, o chinês mandarim e o russo pode custar mais de duas vezes o preço de uma resposta em inglês, e para idiomas como o grego e o hindi, esse valor pode chegar a ser cinco vezes maior.

Essa ferramenta também nos permite visualizar o *ID* (número de identificação) de cada um desses *tokens*. Como esses modelos são baseados em funções matemáticas (como vimos no capítulo anterior), não podemos usar os textos puros nessas funções. Assim, o processo de tokenização também é responsável por mapear cada *token* a um *ID* numérico único, que será posteriormente processado pelas redes neurais. Esse mapeamento entre *tokens* e *IDs* é mantido no **vocabulário** do modelo.

Quando o modelo observa um *token*, este é primeiramente mapeado para um *ID*. Na sequência, esse *ID* é usado para buscar o vetor de *embedding* correspondente àquele *token* e alimentá-lo à rede neural. Ou seja, os *IDs* servem como um índice para encontrar os vetores de *embedding* de cada *token*. Podemos ver os *IDs* para todos os *tokens* da nossa frase em português:

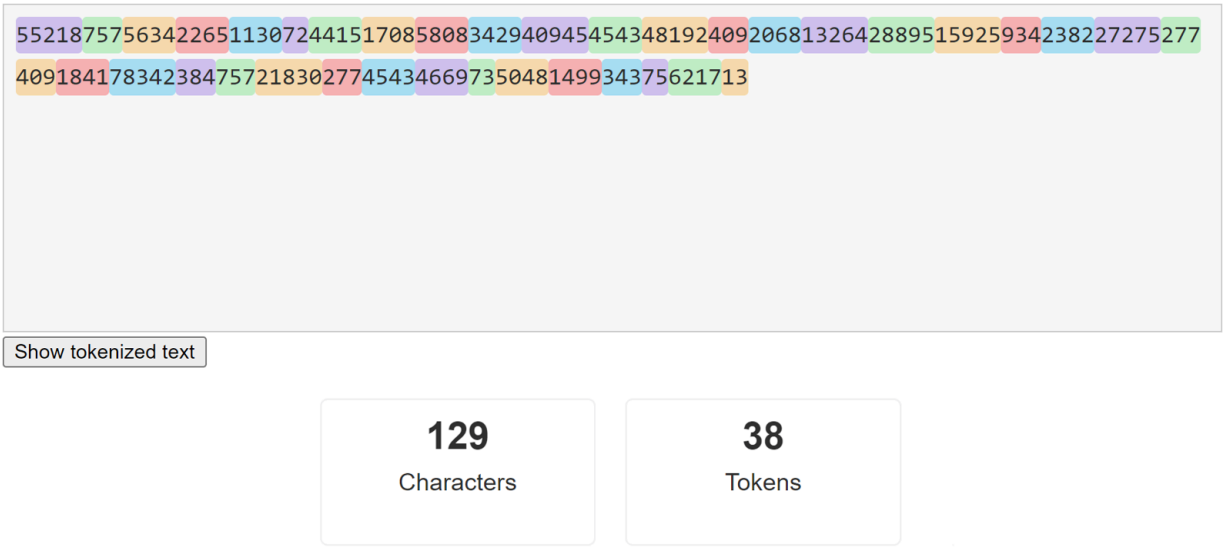

Figura 4.12: IDs do exemplo de tokenização em português utilizando o gpt-tokenizer

Isso demonstra que o *token* " curso" (um caractere de espaço seguido da palavra "curso"), por exemplo, corresponde ao *ID* 48192 no vocabulário do modelo.

4.4 Como o primeiro ChatGPT foi treinado

A criação do *ChatGPT* envolveu inovações na maneira como os modelos de linguagem da *OpenAI* eram treinados, com destaque, principalmente, para uma participação humana mais ativa no processo de treinamento e para o uso de técnicas de *Aprendizado por Reforço* (cuja base vimos no capítulo 2).

Em 2020, a *OpenAI* lançou o modelo *GPT-3*, que apresentou uma enorme evolução em relação a seu predecessor, o *GPT-2*. Apenas para comparação, enquanto o modelo *GPT-2* tinha aproximadamente 1,5 bilhão de parâmetros, o modelo *GPT-3* apresentava 175 bilhões de **parâmetros** (lembrando que, nesse contexto, a palavra "parâmetro" se refere a todos os pesos e vieses individuais dos neurônios que são ajustados durante o processo de treinamento de uma rede neural profunda). Além disso, o *GPT-3* foi treinado com um volume colossal de texto, cerca de 570 GigaBytes no total, tendo sido alimentado de cerca de 300 bilhões de tokens [4] .

Esse tal "volume colossal" de dados com o qual o modelo foi treinado veio majoritariamente da própria internet, o que inclui artigos, sites como a Wikipédia, livros, posts em redes sociais, código etc., a fim de obter um modelo com a capacidade de gerar respostas semelhantes às humanas em diversas áreas do conhecimento. Essa coleta foi feita através de uma técnica chamada *web scraping* (que podemos traduzir livremente como "mineração de dados da web"), que consiste em utilizar robôs para acessarem milhões de páginas da web e extraírem seu conteúdo.

Dataset	Número de tokens	Proporção
Common Crawl	410 bilhões	60%
WebText2	19 bilhões	22%
Books1	12 bilhões	8%
Books2	55 bilhões	8%
Wikipédia	3 bilhões	3%

Figura 4.13: Datasets de dados de treinamento usados para o GPT-3

Um pouco mais tarde, já em 2022, a equipe da *OpenAI* lançou o artigo "*Training language models to follow instructions with human feedback*" (que podemos traduzir livremente como "Treinando modelos de linguagem para seguir instruções com feedback humano")[5] , que dizia que *LLMs* "*podem gerar respostas falsas, tóxicas ou, simplesmente, inúteis para o usuário*" e propunha um refinamento (*fine-tuning* em inglês) do modelo *GPT-3* utilizando *feedback* de seres humanos. A essa proposta de refinamento e consequentes modelos se deu o nome "*InstructGPT*", que se tornou a base para o que seria o *ChatGPT*, a ser lançado oito meses depois.

A ideia dos modelos *GPT-3* e *InstructGPT* era apenas, dado um *prompt* de entrada, completá-lo de maneira coerente. Contudo, como já mencionado, a vantagem do *ChatGPT* em relação a ele e a outros modelos de linguagem mais simples é a sua estrutura em formato de *chat*, como se estivéssemos tendo realmente um diálogo com o modelo, permitindo fazer perguntas e obter respostas coerentes, e o fato de que o *ChatGPT* se lembra do contexto das perguntas e respostas anteriores.

Os treinamentos do *InstructGPT* e do modelo *GPT-3.5*, a ser usado na primeira versão do *ChatGPT*, seguiram as etapas demonstradas no diagrama a seguir:

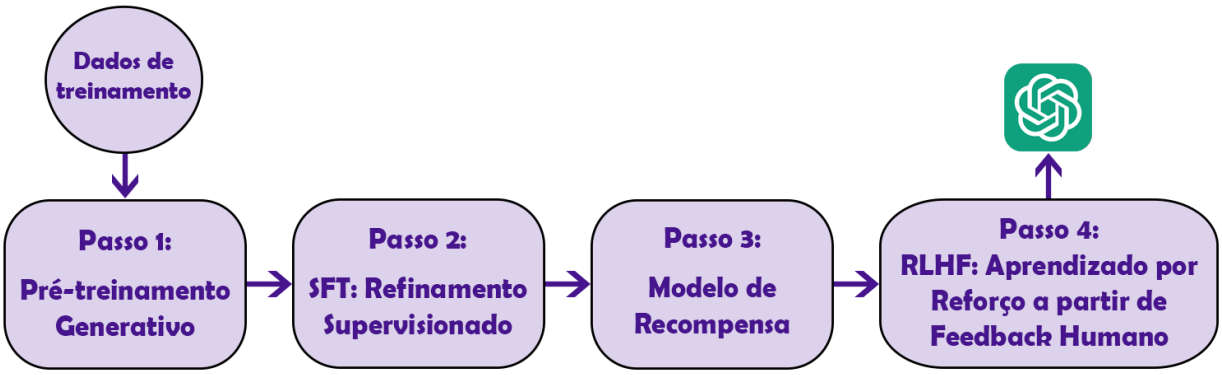

Figura 4.14: Diagrama simplificado demonstrando as etapas no treinamento do InstructGPT e do ChatGPT

Passo 1 — Pré-treinamento Generativo

Nessa etapa, temos todas as letras da sigla *GPT* em ação. Aqui, a rede neural profunda é estruturada utilizando a arquitetura de ***Transformers*** (letra "T"), como vimos no capítulo anterior. Como mencionado anteriormente, o modelo *GPT-3* foi estruturado com 175 bilhões de parâmetros, 96 camadas de atenção e uma janela de contexto de 2.048 tokens, ou seja, ele consegue processar uma entrada de até 2.048 tokens ao mesmo tempo, o que equivale a 3 ou 4 páginas de texto em um arquivo de formato *Word* ou *Google Docs* padrão.

A rede neural começa com valores aleatórios em seus parâmetros, e é feito o **pré-treinamento** (letra "P") inicial do modelo com aquele volume colossal de dados vindos majoritariamente da internet que vimos anteriormente. Esses dados passam por uma limpeza, pelo processo de tokenização e são convertidos em vetores numéricos de *embeddings* para que possam ser processados pelo modelo.

Durante esse pré-treinamento, o papel da rede neural é tentar aprender os padrões dos dados de entrada utilizando *Aprendizado Autossupervisionado*, a fim de ser capaz de **gerar** (letra "G") texto de maneira coerente, completando as entradas oferecidas como exemplo em diferentes contextos.

Mostrando isso de uma forma visual, os 300 bilhões de *tokens* do conjunto de dados seriam usados para gerar pares de entrada e saída para o treinamento da rede neural. Veja a tabela:

Exemplo	Entrada									Saída esperada (rótulo)
#1	O	Chaves	é	um						menino
#2	O	Chaves	é	um	menino					que
#3	O	Chaves	é	um	menino	que				mora
#4	O	Chaves	é	um	menino	que	mora			em
#5	O	Chaves	é	um	menino	que	mora	em		uma
#6	O	Chaves	é	um	menino	que	mora	em	uma	vila

Figura 4.15: Tokens de exemplo e saída esperada

Com uma mesma frase, é possível gerar um número enorme de pares de entrada e saída, que serão apresentados ao modelo (essas frases são apenas um exemplo). O exemplo #1 poderia ser apresentado como uma entrada para que o modelo preveja a saída esperada (a próxima palavra da frase). Obviamente, como ainda está sendo treinado, o modelo vai prever alguma coisa que certamente estará incorreta. A seguir, calculará o erro entre a saída gerada e a esperada (o rótulo na imagem anterior, mas em forma de vetor numérico) e, com esse resultado, a rede neural atualizará seus parâmetros (seus pesos e vieses), a fim de diminuir esse erro nas próximas vezes e fazer previsões melhores. O mesmo será feito para os outros exemplos, e esse processo será repetido milhões e milhões de vezes.

Ao fim desse treinamento, a equipe da *OpenAI* obteve o modelo *GPT-3*, um verdadeiro *LLM*, muito maior que seus predecessores e concorrentes. Dado o tamanho do modelo e a quantidade de *GPUs* (as de alto desempenho usadas para treinar esses modelos) e energia necessárias, é estimado que esse processo de treinamento tenha custado mais de 4 milhões de dólares[6] , com estimativas mais recentes colocando esse valor na casa das dezenas de milhões de dólares. O *GPT-3* era capaz de fazer traduções, responder a perguntas e gerar textos praticamente indistinguíveis daqueles escritos por seres humanos. Porém, as respostas geradas muitas

vezes apresentavam conteúdo tóxico ou enviesado, poderiam difíceis de compreender e não seguiam sempre as instruções dos usuários.

Passo 2 — SFT: Refinamento Supervisionado (*Supervised Fine-Tuning*)

Você se lembra do artigo que eu mencionei que a equipe da *OpenAI* publicou em 2022, com a proposta de refinar (fazer o *fine-tuning*) do modelo *GPT-3*[7] ? Lá, eles explicam como foi feito esse processo de refinamento do *GPT-3*, e um dos pontos mais interessantes (e controversos) é que ele teve grande participação de seres humanos. Foram contratadas 40 pessoas para rotular os dados de maneira confiável. Dados rotulados nos lembram de que? **Aprendizado Supervisionado**, é claro!

Primeiramente, a equipe coletou *prompts* reais de usuários do *GPT-3*. A fim de maximizar a diversidade no conjunto de dados, um usuário do *GPT-3* poderia ter no máximo 200 *prompts* selecionados, *prompts* que começassem de maneira muito similar eram deduplicados (ou seja, apenas um deles era mantido) e todas as informações pessoais foram removidas. Depois dessa coleta e limpeza, foi pedido aos rotuladores que criassem *prompts* de categorias que não tivessem sido muito abordadas pelos *prompts* reais de usuários.

Na sequência, os rotuladores passaram a escrever respostas apropriadas para cada um dos *prompts* coletados da maneira que eles (e a equipe da *OpenAI*) gostariam que o modelo respondesse àqueles *prompts*, sempre evitando usar linguagem ou conteúdo ofensivos ou preconceituosos. 96% desse conteúdo era na língua inglesa. Ao final, chegaram a um conjunto de dados rotulados com 13.000 pares de *prompt*-resposta, que foram usados para realizar o refinamento (*fine-tuning*) do modelo *GPT-3* inicial através de *Aprendizado Supervisionado*. O refinamento foi treinado por 16 épocas e, como resultado, gerou o modelo que foi chamado de

GPT-3.5 (também chamado de "*modelo SFT*"), a primeira versão do modelo que seria disponibilizado no lançamento do *ChatGPT*.

Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.

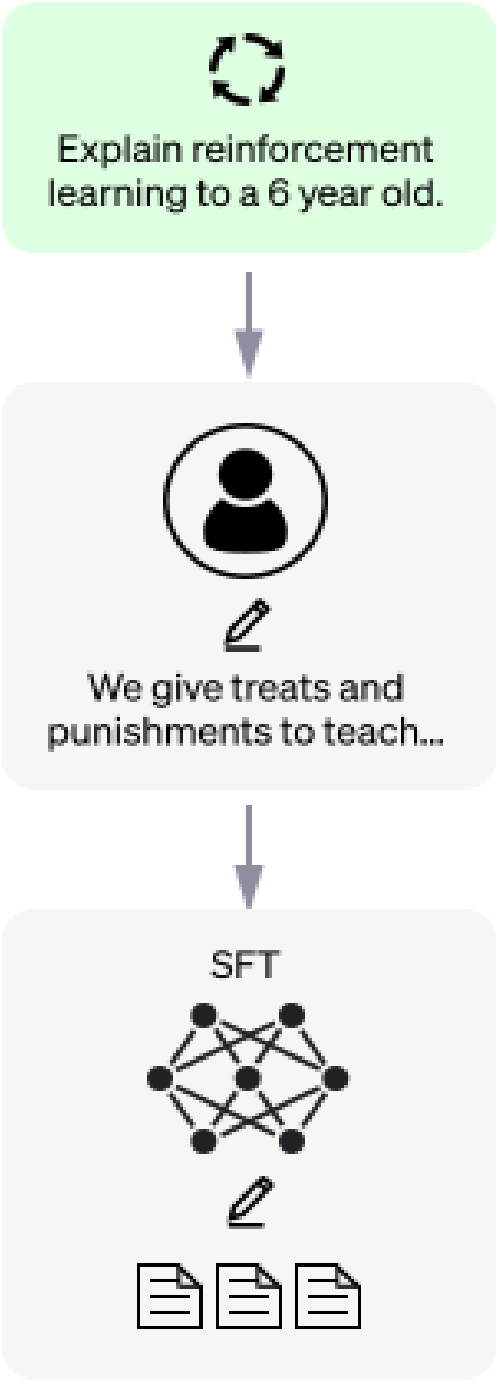

Figura 4.16: Etapa de Refinamento Supervisionado de openai.com/blog/chatgpt

Passo 3 — Modelo de Recompensa (*Reward Model*)

O Passo 2 foi só o começo! Depois de passar por esse primeiro refinamento, o modelo já começou a fornecer respostas em um formato melhor e mais coerentes com os *prompts* pedidos. Aqui no Passo 3, a ideia era treinar o Modelo de Recompensa que seria usado para a etapa de *Aprendizado por Reforço* no Passo 4.

Inicialmente, vários *prompts* foram selecionados, e o modelo *SFT* obtido ao final do refinamento do passo anterior foi usado como base para gerar de 4 a 9 respostas para cada um desses *prompts*. Essas 4 a 9 respostas eram, então, analisadas pelos rotuladores e ranqueadas da melhor para a pior. Pense na paciência dessas pessoas. Ler nove versões ligeiramente diferentes da mesma resposta e ter que decidir qual é a 'melhor'. É o tipo de tarefa que faria qualquer um questionar as próprias escolhas de vida, mas foi fundamental para treinar o 'bom gosto' do modelo.

Como um mesmo rotulador classificava múltiplas respostas, era possível gerar múltiplas comparações de "resposta melhor/pior". Por exemplo, se um rotulador ranqueasse 4 respostas A, B, C e D na ordem "C > B > A > D", automaticamente teríamos 6 pares de "resposta melhor/pior" ranqueados: (C > B), (C > A), (C > D), (B > A), (B > D) e (A > D).

Com esses *prompts* e respostas ranqueadas em mãos, foi possível treinar o Modelo de Recompensa, que tinha como entrada um *prompt* e as respostas ranqueadas, e como saída, uma pontuação chamada de **recompensa**.

Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

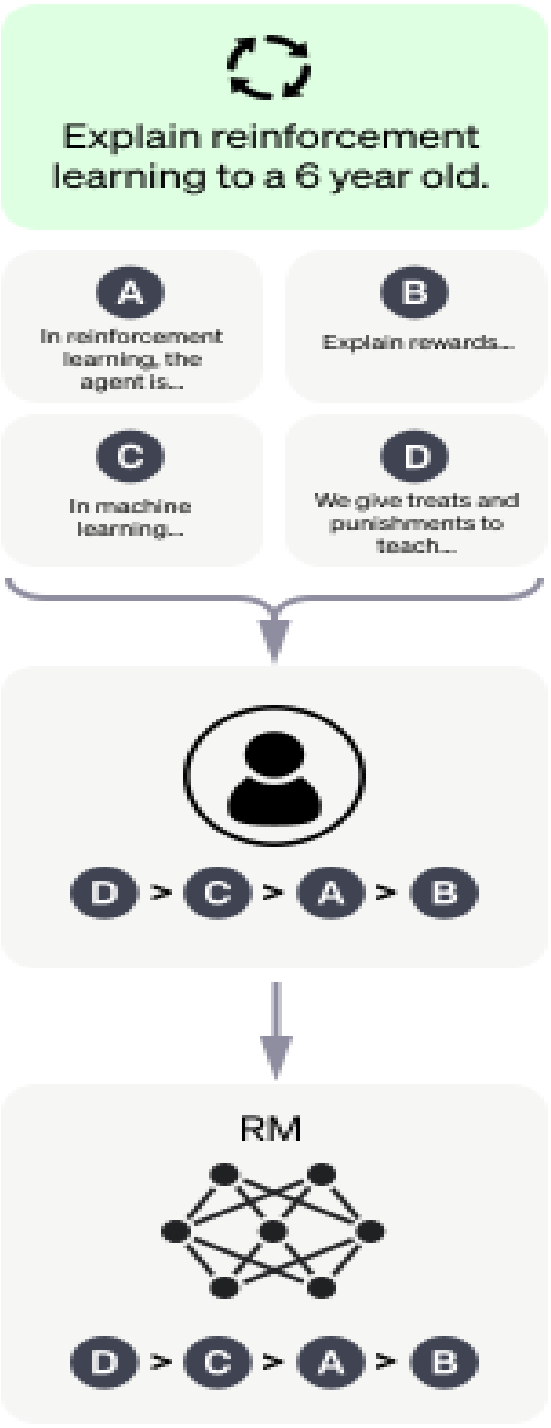

Figura 4.17: Etapa de Modelo de Recompensa de openai.com/blog/chatgpt

Passo 4 — RLHF: Aprendizado por Reforço a partir de Feedback Humano (*Reinforcement Learning from Human Feedback*)

Nessa etapa final, o modelo *SFT* continuou a ser treinado. Para tal, foi usado um algoritmo de **Aprendizado por Reforço** chamado "Otimização de Política Proximal" (de sigla ***PPO*** em inglês, de "*Proximal Policy Optimization*")[8] , lançado pela própria equipe da *OpenAI* em 2017 e muito comum em *LLMs*.

A "política" (lembra desse termo? Vimos sobre ele no capítulo 2. É a estratégia que o agente seguirá para tomar decisões) desse algoritmo de *Aprendizado por Reforço* era maximizar a recompensa, ou seja, o objetivo era gerar respostas que aumentassem a sua recompensa de acordo com o Modelo de Recompensa obtido no Passo 3.

Nesse processo, um *prompt* inédito do conjunto de *prompts* dos usuários ou rotuladores é enviado para o modelo, e uma resposta é gerada para ele. Essa resposta é, então, enviada para o Modelo de Recompensa obtido no Passo 3, a fim de ser ranqueada em termos de qualidade em relação ao *prompt* em questão. Após passar pelo Modelo de Recompensa, é gerada uma pontuação de recompensa para esse par *prompt*-resposta, a ser usada pelo algoritmo de *PPO* para refinar os parâmetros do modelo e atualizar a política do algoritmo de *PPO*.

Os passos 3 e 4 podem ser iterados repetidamente, a fim de refinar ainda mais o modelo.

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

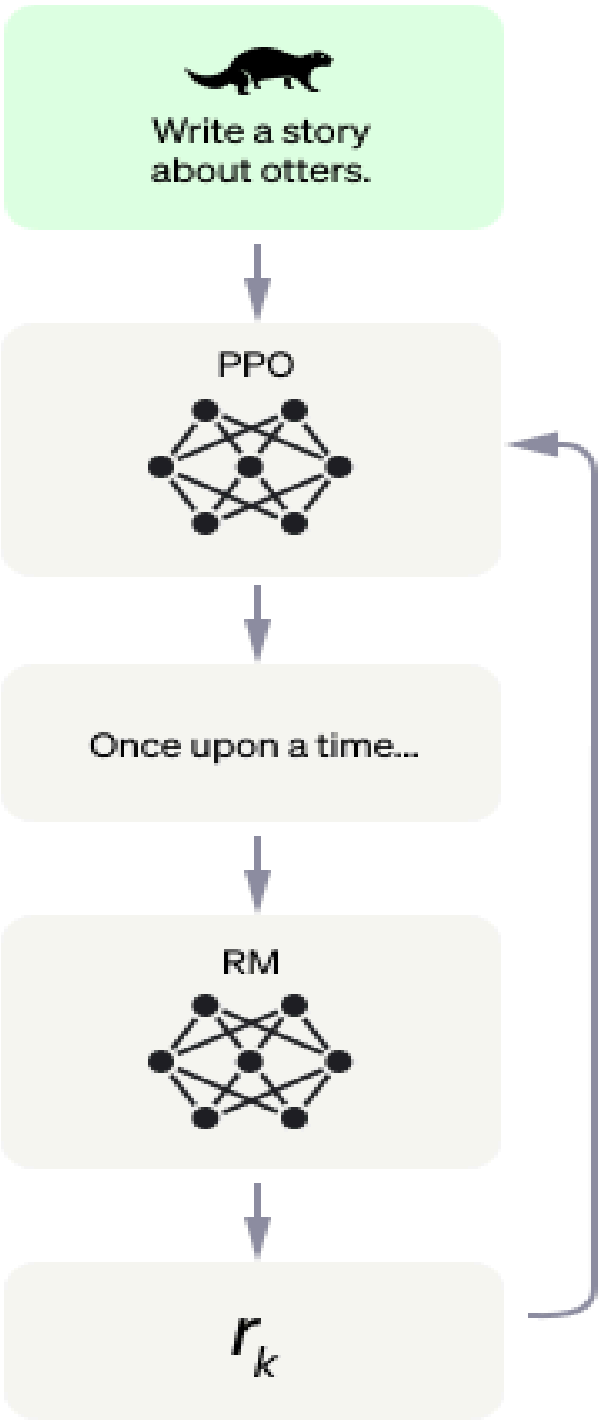

Figura 4.18: Etapa de Aprendizado por Reforço a partir de Feedback Humano de openai.com/blog/chatgpt

Resumindo o processo: primeiro, ensinaram o modelo com exemplos (Passo 2). Depois, treinaram um 'juiz' para saber o que é uma boa resposta (Passo 3). Por fim, fizeram o modelo original gerar respostas incansavelmente até que o 'juiz' ficasse satisfeito (Passo 4). É basicamente um *reality show* de talentos para IAs, mas com muito mais matemática e menos choro.

Ao final desse processo, a equipe da *OpenAI* chegou ao modelo que chamaram de *InstructGPT*, o irmão mais velho do *GPT-3.5*, que apresentava grandes evoluções em comparação com o modelo *GPT-3*, como respostas mais coerentes, confiáveis, informativas e menos tóxicas. Além disso, ele era muito melhor em seguir instruções do usuário do que o *GPT-3* sem refinamento.

Diferenças no treinamento do InstructGPT e do ChatGPT

Conforme mencionado anteriormente, o *InstructGPT* foi desenhado para seguir as instruções dadas em um *prompt* e respondê-lo ou completá-lo de maneira coerente. Ao ver o quão bem ele funcionava (em comparação com todos os outros *LLMs* da época), a equipe da *OpenAI* pensou: "*E se nós pudéssemos ter uma conversa natural com o InstructGPT, onde ele se lembrasse do contexto de todas as mensagens anteriores?*". *Chatbots* não eram uma tecnologia nova, mas um *chatbot* com o poder matemático de raciocínio do *InstructGPT* poderia ser algo muito inovador. Quem sabe, até disruptivo, não é mesmo?

Os detalhes por trás do treinamento do *ChatGPT* não são tão abertos quanto os do *InstructGPT*, mas a *OpenAI* forneceu algumas explicações sobre como esse processo foi feito[8] . Basicamente, o método de treinamento tem a mesma estrutura e passos que os usados para o *InstructGPT*, com algumas diferenças principalmente no conjunto de dados usados para o refinamento (*fine-tuning*).

No Passo 2, o *fine-tuning* não é baseado apenas em rotular uma resposta ideal a um *prompt* dado. Aqui, os rotuladores eram responsáveis por gerar conversas inteiras, onde um deles fazia o papel do usuário e o outro atuava como se fosse o *chatbot*, interagindo e respondendo de forma coerente e correta. Isso gerou vários "históricos de conversa". Além disso, parte do conjunto de dados do próprio *InstructGPT* foi convertida em formato de diálogo e, junto aos históricos gerados pelos rotuladores, foi usada para formar o conjunto de dados de refinamento e realizar o refinamento através de *Aprendizado Supervisionado*.

No Passo 3, era necessário, primeiramente, coletar dados de comparação, que consistiam em duas ou mais respostas geradas pelo modelo, ranqueadas quanto à sua qualidade. Essa coleta foi feita pegando diversas conversas que os rotuladores haviam tido com o *chatbot*, selecionando aleatoriamente uma resposta gerada pelo modelo, buscando várias respostas alternativas e pedindo aos rotuladores que as ranqueassem da melhor para a pior. Com isso, foi treinado o Modelo de Recompensa, onde a entrada era um histórico de conversa com uma resposta gerada pelo modelo, e a saída, uma pontuação correspondente, com base nas classificações criadas pelos rotuladores.

Ao final, assim como no *InstructGPT*, o modelo é refinado com o algoritmo *PPO* de *Aprendizado por Reforço* em um processo de múltiplas iterações. O resultado disso foi o famoso modelo *GPT-3.5* disponibilizado no *ChatGPT*, lançado de forma oficial ao público em novembro de 2022, que nada mais é do que uma versão de um dos modelos de linguagem da série *GPT-3.5*, mas refinada para o uso em conversas, e que surpreendeu até mesmo a equipe da *OpenAI*, dada a dimensão da melhora produzida por esses *fine-tunings*.

Para esclarecer, a série de modelos *GPT-3.5* inclui o modelo *SFT* que vimos ser gerado no Passo 2 como um refinamento do *GPT-3*, mas também o modelo *InstructGPT* e suas sub-versões e, claro, o modelo otimizado para conversas utilizado na versão pública e gratuita do *ChatGPT* na época do seu lançamento (e suas atualizações, claro). A

janela de contexto do *GPT-3.5* foi aumentada para 4.096 *tokens* e, posteriormente, foi lançada uma versão com 16.385 *tokens*. Seu treinamento, inicialmente, possuía dados de apenas até setembro de 2021 (isso quer dizer que, para perguntas feitas sobre fatos ocorridos depois dessa data, o modelo não teria as informações, e provavelmente inventaria alguma **alucinação** — falaremos mais sobre alucinações em breve).

Esse treinamento foi feito inicialmente usando uma infraestrutura de supercomputadores da *Microsoft Azure* com *GPUs* da *NVIDIA*. Essa infraestrutura foi construída pela *Microsoft* exclusivamente para a *OpenAI*, por ser uma das empresas investidoras por trás dela e, segundo o vice-presidente de IA e Cloud da *Microsoft*, o projeto teria custado centenas de milhões de dólares [9] .

Analisando ainda o funcionamento do *ChatGPT* e considerando a rede neural do modelo *GPT-3.5* que, como vimos, possuía aproximadamente 175 bilhões de parâmetros, isso significa, essencialmente, que, cada vez que o *GPT-3.5* é colocado em ação para gerar um novo *token*, o modelo realiza uma série de cálculos com os *embeddings* que, ao longo do processo, envolverão todos esses parâmetros. Claro, isso é feito de maneira distribuída e paralela entre várias camadas, passando os resultados desses cálculos dos neurônios para as camadas subsequentes, mas, ainda assim, é surpreendente a velocidade com a qual a ferramenta consegue produzir um texto completo e coerente (muito graças a esses enormes *clusters* de *GPUs*).

Em modelos mais recentes, existem arquiteturas como *Mixture-of-Experts (MoE)*, graças as quais os modelos não precisam passar por todos os parâmetros da rede neural para calcular o próximo *token* (teremos uma seção específica para a arquitetura *Mixture-of-Experts*).

Desde o seu lançamento em novembro, o modelo do *GPT-3.5* do *ChatGPT* foi aprimorado e atualizado diversas vezes. Um dos pontos principais de foco foi e tem sido para evitar *jailbreaking*, que se trata

do ato de um usuário tentar engenhar o *prompt* de maneira a tentar enganar o treinamento do modelo, a fim de que ele responda de uma maneira indesejada pela empresa *OpenAI*, seja dando respostas tóxicas, impróprias ou perturbadoras. Para fortalecer o modelo contra esse tipo de ataque, a *OpenAI* tem usado uma técnica de *Machine Learning* chamada "Treinamento Adversário" (*Adversarial Training*, em inglês), que consiste em ter dois *chatbots* conversando, um atuando como atacante e tentando gerar esses *jailbreaks*, e o outro respondendo da maneira que foi treinado normalmente. Quando um ataque é bem-sucedido, seus dados são coletados e, com isso, um novo conjunto de dados de treinamento é criado, a fim de refinar o modelo para que não caia mais neles no futuro.

Porém, *jailbreaking* não é o único problema que o *ChatGPT* encontrou.

4.5 Alucinações

Ao ler sobre o *ChatGPT* e outros *LLMs* na teoria, é comum ter um sentimento de admiração sobre toda a ciência e tecnologia por trás de seu funcionamento. Porém, na prática, nem tudo é perfeito. Apesar de seu treinamento intenso e sofisticado, esses modelos, vez ou outra, cometem erros fascinantes.

Ao perguntar a um modelo *"Qual é a capital da Austrália?*", ele calculará as possíveis opções de resposta com base em seu treinamento, cada uma com uma probabilidade associada, como já vimos anteriormente. Em seu *embedding*, é provável que a cidade de "Canberra" esteja próxima aos conceitos de "Austrália" e "capital", já que é a verdadeira capital desse país e, por isso, o modelo a proponha com uma probabilidade maior. Porém, lembre-se

de que o modelo é probabilístico e sempre oferece algumas escolhas. Nesse caso, o modelo poderia oferecer também as cidades de "Sydney", "Melbourne" e "Brisbane" como opções, já que também são cidades importantes da Austrália. E, exatamente por funcionar dessa forma, dependendo do valor da temperatura ou simplesmente de escolhas estocásticas (como vimos, mesmo a temperatura 0 permite variações na resposta), o modelo poderia retornar uma dessas outras cidades como resposta, o que nós, seres humanos que conhecemos a resposta correta, veríamos como um erro crasso.

Outro exemplo foi dado pelo jornal "*The New York Times*" [10] , onde um de seus jornalistas perguntou ao *ChatGPT*: "*Qual foi o primeiro artigo que o The New York Times escreveu sobre Inteligência Artificial?*" e obteve a resposta: "*Em 10 de julho de 1956, em um artigo intitulado 'As máquinas serão capazes de aprender e resolver problemas, cientistas preveem' sobre uma conferência seminal no Dartmouth College.*". Essa conferência realmente existiu, mas o artigo, nunca. O *ChatGPT* simplesmente inventou essa informação.

No dialeto da área de PLN, esse tipo de erro é chamado de "**alucinação**", ou seja, a geração de informações que, embora esteticamente plausíveis, estão totalmente desconectadas da realidade. Para colocar de uma forma mais simples, modelos de linguagem como o *ChatGPT*, *Gemini* e *Claude* podem gerar coisas que dão a impressão de estarem corretas, mas que são completamente fictícias.

Além disso, por ser um hábil escritor, esses modelos conseguem gerar essas respostas de forma confiante e convincente, o que pode ter sérias consequências. Ele pode, em alguns casos, acabar disseminando informações errôneas, enganosas ou simplesmente prejudiciais, por exemplo, para um usuário que busque respostas a perguntas sobre casos de saúde, jurídicas ou sobre fatos relevantes.

Um usuário desses *LLMs* pode utilizar conceitos de Engenharia de *Prompt* para tentar facilitar a compreensão do modelo sobre a

pergunta e, quem sabe, obter respostas mais claras e corretas, mas nem sempre isso resolverá as alucinações.

Esses modelos de linguagem são treinados em grandes quantidades de dados de texto e aprendem a gerar saídas baseando-se nesses dados, mas, no geral, não têm em si a capacidade de verificar a veracidade das informações retornadas levando em conta apenas o seu treinamento. Se eles não estão conectados à internet, todas as suas respostas dependem do que foi aprendido durante o treinamento e de suas escolhas probabilísticas ao gerá-las. Por esse motivo, uma das possíveis causas de alucinações pode ser desinformação no próprio conjunto de dados de treinamento do modelo.

Ainda não existe uma única solução para resolver esse problema, mas existem estratégias que as empresas usam para tentar mitigá-lo, como melhorar a qualidade e veracidade factual dos dados de treinamento através de uma curadoria e rotulamento mais minuciosos (o que pode ser muito difícil e custoso para quantidades tão grandes de dados); conectar o sistema à internet para buscar em páginas confiáveis uma referência para a resposta gerada (o que, por si só, também não garante a veracidade dos dados e pode gerar outros problemas); manter uma temperatura mais baixa quando estiver buscando dados que necessitem de uma precisão maior; e refinar cada vez mais o modelo com *Aprendizado por Reforço a partir de Feedback Humano* (*RLHF*) e outras técnicas.

No entanto, até que tenhamos soluções definitivas (se as tivermos algum dia), é importante manter sempre um olhar crítico sobre as respostas geradas. Não assuma que elas estejam corretas simplesmente por parecerem coerentes ou bem escritas e faça sempre a verificação posterior em fontes confiáveis.

4.6 GPT-4 e GPT-4o

O modelo *GPT-4*, lançado em março de 2023, foi o sucessor direto dos modelos da série *GPT-3.5*. Ele é um modelo multimodal, ou seja, que aceita mais de uma modalidade de dado como entrada — o *GPT-4* aceitava texto e imagens como entrada. Dois modelos foram liberados, um com janela de contexto de 8.192 *tokens* e outro com 32.768. Seu treinamento, assim como o de seu predecessor, possuía dados apenas até setembro de 2021.

Para testar as suas capacidades, o *GPT-4* foi posto à prova com vários exames desenvolvidos para humanos, nos quais exibiu um desempenho excelente, muitas vezes obtendo uma nota melhor do que a grande maioria dos examinados humanos. Por exemplo, em um exame simulado da ordem de advogados, o *GPT-4* conseguiu uma pontuação que o colocou entre os 10% dos melhores examinados. Para efeito de comparação, o modelo *GPT-3.5*, quando testado com o mesmo exame, pontuou entre os 10% piores [11] .

O modelo também foi testado em diversos *benchmarks* específicos para modelos de *Machine Learning* e superou todos os *LLMs* existentes até então em praticamente todos os quesitos, inclusive o modelo *GPT-3.5*.

Benchmark	**GPT-4** Evaluated few-shot	**GPT-3.5** Evaluated few-shot	**LM SOTA** Best external LM evaluated few-shot	**SOTA** Best external model (includes benchmark-specific training)
MMLU Multiple-choice questions in 57 subjects (professional & academic)	86.4% 5-shot	70.0% 5-shot	70.7% 5-shot U-PaLM	75.2% 5-shot Flan-PaLM
HellaSwag Commonsense reasoning around everyday events	95.3% 10-shot	85.5% 10-shot	84.2% LLAMA (validation set)	85.6% ALUM
AI2 Reasoning Challenge (ARC) Grade-school multiple choice science questions. Challenge-set.	96.3% 25-shot	85.2% 25-shot	84.2% 8-shot PaLM	85.6% ST-MOE
WinoGrande Commonsense reasoning around pronoun resolution	87.5% 5-shot	81.6% 5-shot	84.2% 5-shot PALM	85.6% 5-shot PALM
HumanEval Python coding tasks	67.0% 0-shot	48.1% 0-shot	26.2% 0-shot PaLM	65.8% CodeT + GPT-3.5
DROP (f1 score) Reading comprehension & arithmetic.	80.9 3-shot	64.1 3-shot	70.8 1-shot PaLM	88.4 QDGAT

Figura 4.19: Comparação do GPT-4 com outros modelos em grandes benchmarks de https://openai.com/research/gpt-4

Diferentemente dos modelos anteriores, a arquitetura do *GPT-4* foi mantida em segredo pela *OpenAI*. O lançamento do *ChatGPT* com o

GPT-3.5 no ano anterior causou um momento de disrupção no mundo da tecnologia e também junto ao público geral, o que levou algumas das maiores empresas do planeta como o *Google*, a *Meta* e a *Amazon* a correrem atrás do prejuízo para não ficarem para trás na corrida da IA. Tanto o *Google* quanto a *Meta* já trabalhavam há anos em modelos *LLM* similares, mas longe de estarem prontos para serem apresentados ao público. Assim, ter o seu melhor modelo como um "segredo de estado" foi uma estratégia de mercado adotada pela *OpenAI* para continuar à frente dos concorrentes.

Porém, com o passar dos meses, vários rumores vieram à tona sobre a arquitetura do *GPT-4*. Acredita-se que ele possuía 1,8 trilhão de parâmetros (os pesos e vieses da rede neural) e era uma rede de 120 camadas de profundidade [12] , mais de 10 vezes mais do que os 175 bilhões de parâmetros do *GPT-3* e *GPT-3.5*. De acordo com as poucas informações liberadas pela *OpenAI*, o treinamento do modelo foi feito de maneira muito similar a essa explicada acima para o *GPT-3.5*, mas com uma recompensa adicional de segurança na fase de *RLHF*, a fim de reduzir a quantidade de saídas potencialmente nocivas[11] . Perguntado sobre o custo do treinamento do modelo, Sam Altman, o *CEO* da *OpenAI*, respondeu que teria custado mais de 100 milhões de dólares.[13]

Existe também um rumor não confirmado pela *OpenAI* de que o *GPT-4* não seria um modelo denso e monolítico como o *GPT-3* e o *GPT-3.5*, mas sim que utilizaria uma arquitetura de *Mixture-of-Experts* (que será explicada detalhadamente em breve), combinando 8 partes ou submodelos de 220 bilhões de parâmetros cada, totalizando esses ~1,8 trilhões de parâmetros, e que cada um deles seria refinado para uma determinada especialidade.[14]

Como mencionado, o *GPT-4* e seus sucessores são modelos multimodais e conseguem reconhecer imagens, o que, ligado a seus motores de texto, são capazes de demonstrar um grande poder de reconhecimento de coisas do mundo real, mesmo conceitos mais abstratos.

Um exemplo de uso dessas capacidades visuais que ficou muito popular na internet foi o seguinte: dada a imagem a seguir, perguntou-se ao *GPT-4*: "*O que você acha que essa imagem significa?*".

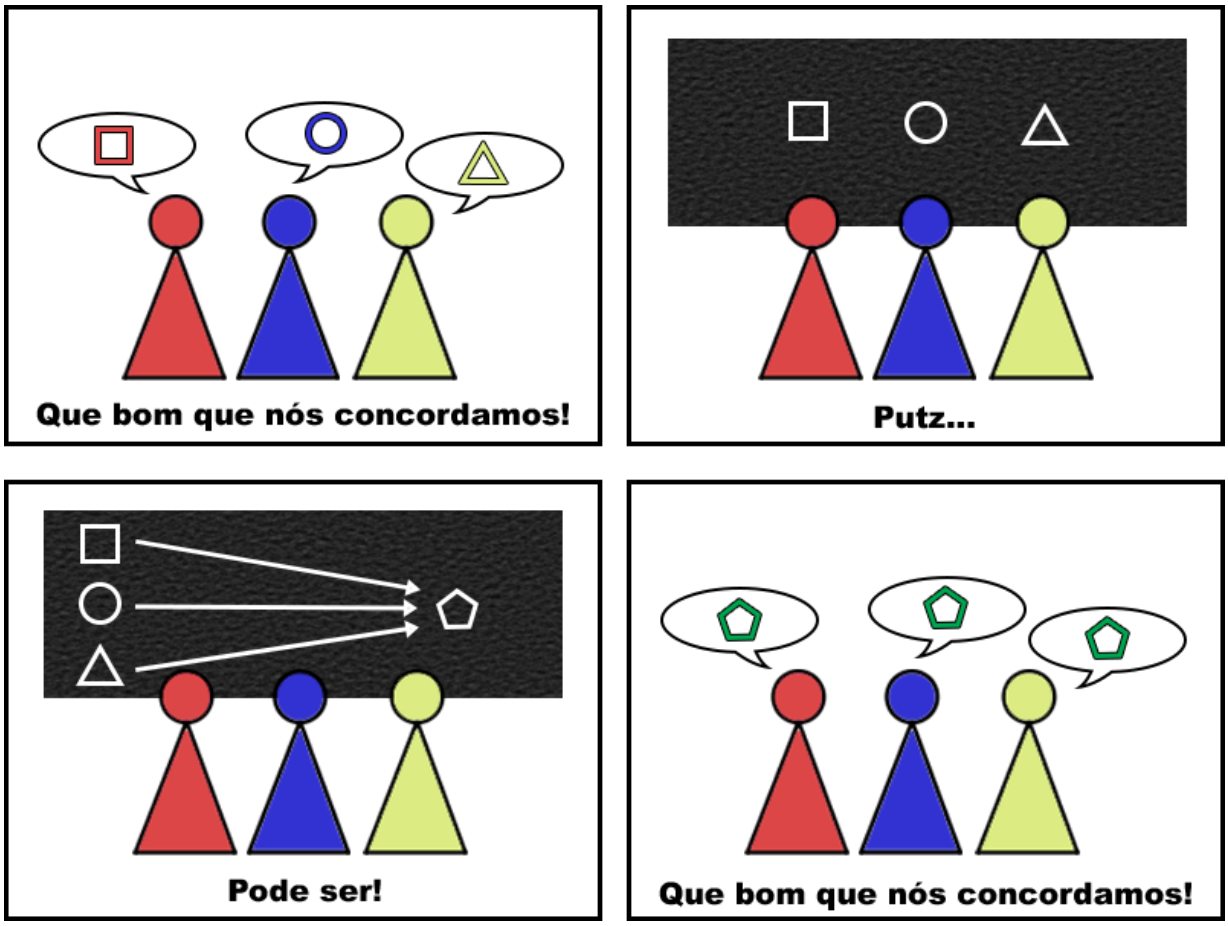

Figura 4.20: Reprodução de imagem de uma tirinha com pessoas achando erroneamente que concordavam e, depois, chegando a um acordo

Sabe o que o modelo respondeu?

A imagem retrata o conceito de dinâmicas e perspectivas de grupo. Especificamente:

1. No primeiro painel, cada indivíduo tem uma forma diferente em mente, mas parece pensar que estão pensando na mesma coisa, conforme indicado pelo texto "Que bom que nós concordamos".

2. No segundo painel, quando a forma real é revelada, fica evidente que nenhum deles estava pensando na mesma coisa, levando a um momento de compreensão com a legenda "Putz..."

3. No terceiro painel, as figuras são mostradas novamente com formatos diferentes, mas desta vez há um momento de "Pode ser!", indicando uma percepção ou percepção repentina.

4. O último painel mostra que, após alguma discussão ou reflexão, todos chegaram a um consenso ou entendimento compartilhado, pois todos visualizam a mesma forma e a legenda reafirma "Que bom que nós concordamos".

No geral, parece destacar a importância da comunicação, compreensão e alinhamento em ambientes de grupo. Sugere que, mesmo que os indivíduos pensem que estão alinhados, sem uma comunicação clara, podem ocorrer mal-entendidos. Mas com esforço e discussão, um entendimento partilhado pode ser alcançado.

Impressionante, não? Ele conseguiu não apenas ler a imagem, mas também a interpretou, sendo capaz de perceber conceitos muito abstratos que, até pouquíssimo tempo atrás, apenas seres humanos seriam capazes de perceber.

Além disso, no mesmo mês do lançamento, a *Microsoft* anunciou que seu buscador, o *Bing*, já estava funcionando alimentado por uma

versão do modelo *GPT-4* personalizada para buscas[15] . Essa conexão de um modelo de IA tão poderoso com um buscador gerou outro furor no mercado e no mundo da tecnologia e um questionamento geral no sentido de: "*onde mais poderemos conectar esses modelos?*". A verdade é que ainda estamos vivenciando essa mudança dia após dia e vendo novas ferramentas que usam IA serem lançadas o tempo todo.

Dado o momento de escrita deste livro, o *GPT-4* já foi aposentado pela *OpenAI*. Foi lançado em maio de 2024 o modelo *GPT-4o* (a letra "o" é uma abreviação de "*omni*"), também multimodal e bastante superior ao *GPT-4* nos principais *benchmarks*. Ele conta com uma janela de contexto de 128.000 *tokens* e, diferentemente do *GPT-3.5* e do *GPT-4*, que dependiam de outros modelos para processar áudio e imagens, o *GPT-4o* oferece suporte nativo à conversa por voz e à geração de imagens a partir de *prompts*.

Em 25 de março de 2025, a OpenAI lançou um modelo de geração de imagens nativo do GPT-4o chamado de *GPT-Image-1*, como sucessor do *DALL-E 3*, que rapidamente viralizou nas redes sociais, principalmente com usuários recriando memes e fotos pessoais no estilo do *Studio Ghibli*, famoso estúdio japonês de animações como "A Viagem de Chihiro" e "Meu Vizinho Totoro". Posteriormente, a *OpenAI* afirmou a ferramenta foi usada por mais de 130 milhões de usuários e que foram criadas mais de 700 milhões de imagens com o *GPT-Image-1* apenas na primeira semana[16] .

Figura 4.21: Imagem de um meme brasileiro gerada pelo GPT-4o no estilo Studio Ghibli

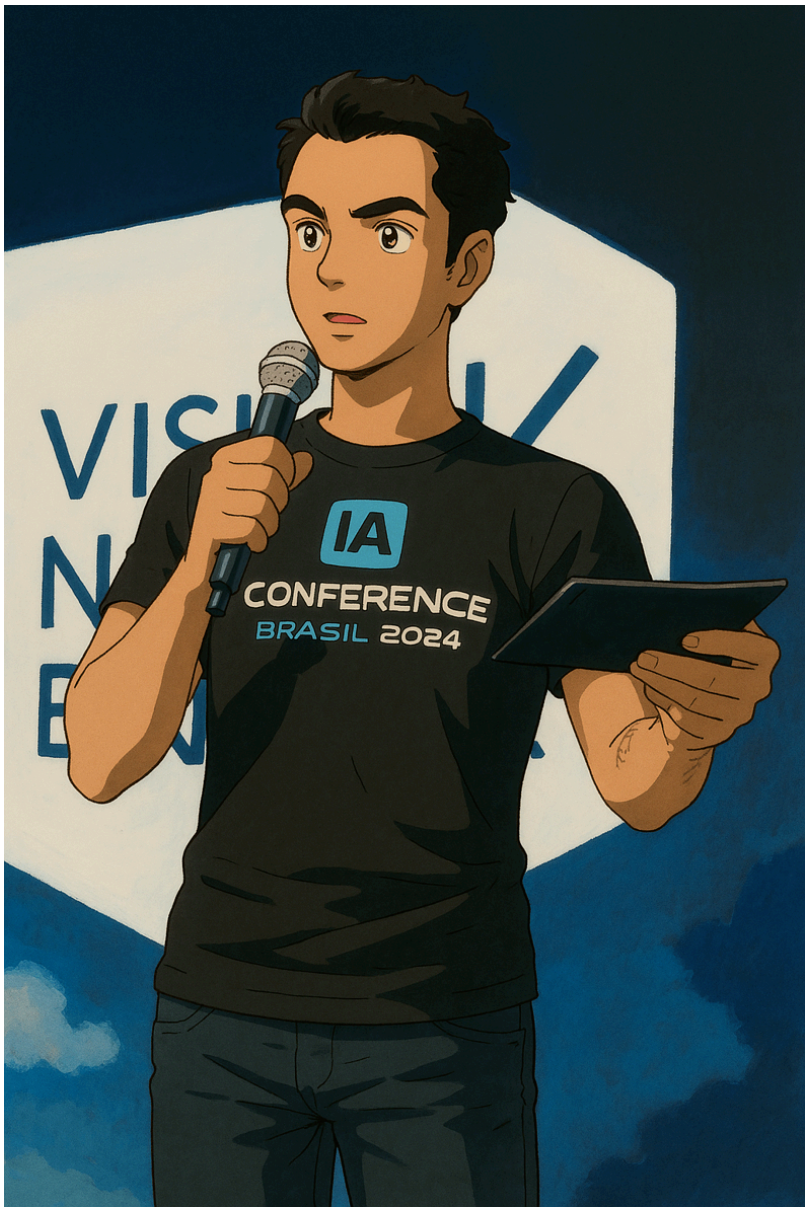

Figura 4.22: Imagem do Fabrício Carraro como anfitrião do IA Conference Brasil 2024 no estilo Studio Ghibli

4.7 Mixture-of-Experts (MoE)

O rumor que comentei anteriormente sobre o GPT-4 talvez estar usando uma arquitetura de *Mixture-of-Experts (MoE)* ("Mistura de Especialistas" em português) acabou se mostrando um vislumbre de

uma das tendências mais importantes para a construção de modelos de linguagem de ponta. Embora a *OpenAI* nunca tenha confirmado oficialmente essa informação, diversos pesquisadores da área acreditam que o modelo realmente segue esse paradigma, afinal, construir um único modelo monolítico com 1,8 trilhão de parâmetros (como é estimado no caso do GPT-4) seria algo extremamente custoso e ineficiente para treinar e, principalmente, para rodar a inferência em produção. Por outro lado, temos a confirmação de que os modelos mais recentes da família *Gemini* do *Google*[17] , vários modelos da francesa *Mistral AI* e os modelos *open source* chineses *DeepSeek* da *DeepSeek AI* e *Kimi K2* da *Moonshot AI* utilizam, sim, esta arquitetura.

Mas o que é, afinal, essa tal de arquitetura *Mixture-of-Experts*?

Imagine um hospital. Lá, não temos um único médico extremamente especialista em todas as áreas possíveis da medicina, de cardiologia a dermatologia, passando por neurologia. Hospitais têm uma equipe de especialistas, cada um com grande foco em sua especialidade. Quando um paciente chega, existe uma triagem do problema, e o paciente é encaminhado para o especialista mais adequado. Se o caso for complexo, talvez dois ou três especialistas precisem colaborar. O ponto principal é que, para tratar uma dor de garganta, você não precisa mobilizar a neurocirurgiã-chefe.

A arquitetura *Mixture-of-Experts* funciona de maneira análoga. Em vez de construir um único modelo neural gigante e "denso", onde todos os trilhões de parâmetros teriam que ser ativados para gerar cada *token* da resposta (o que seria equivalente a ter um único médico superespecialista em todas as áreas da medicina ao mesmo tempo), a abordagem *MoE* transforma isso em um modelo "esparso", construindo vários submodelos menores, os "especialistas", e junto a eles uma pequena rede neural chamada de "*gating network*" ("rede de roteamento" em português). Essa rede funciona como a pessoa da triagem do hospital que, estressada,

olha para o seu prompt e decide: "Isso parece um problema de programação, chame o Dr. Python. Ah, você quer um um texto criativo? Mande para a Dra. Lispector".

Para cada *token* que o modelo precisa processar, a rede de roteamento analisa a informação e decide dinamicamente qual ou quais especialistas são os mais qualificados para a tarefa. Tipicamente, apenas um pequeno número de especialistas (por exemplo, 1 ou 2 de um total de 8) é ativado.

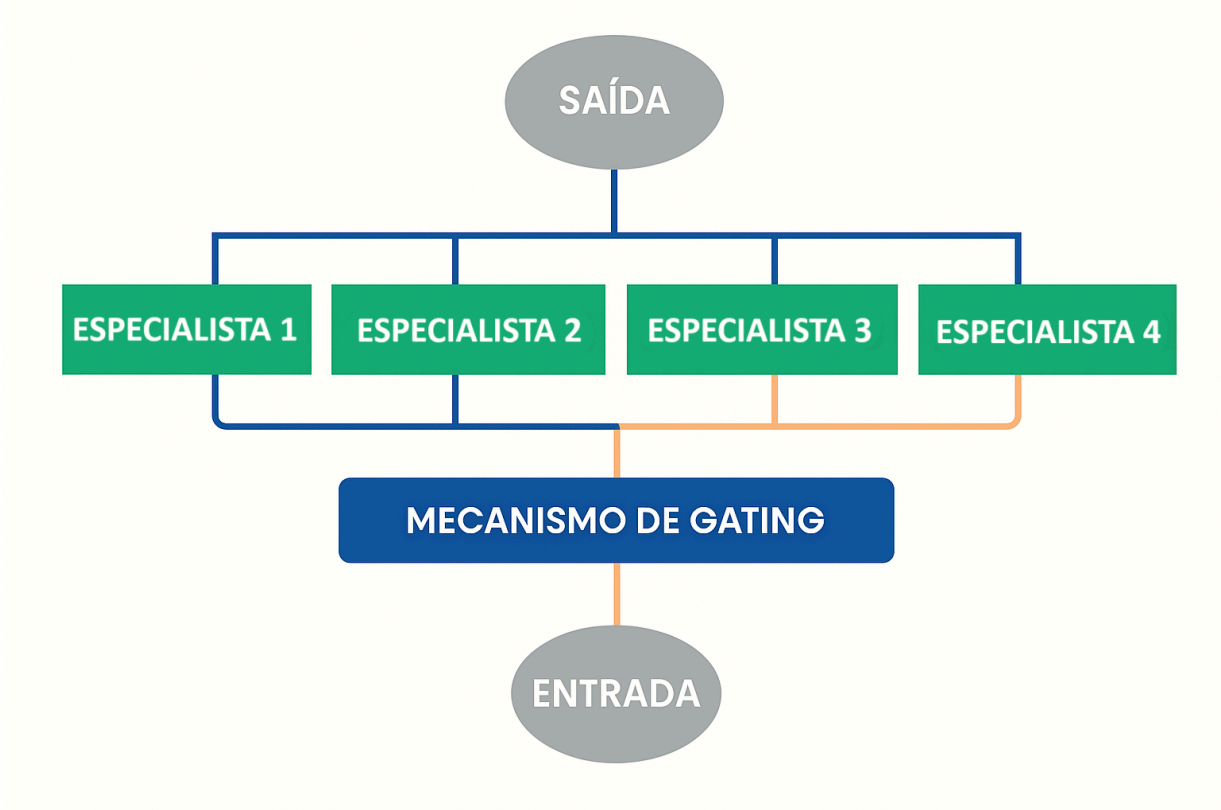

Figura 4.23: Representação do funcionamento da arquitetura Mixture-of-Experts

A corrida por modelos cada vez maiores e mais capazes esbarrou em um limite prático: o custo computacional e energético de treinar e operar esses gigantes. Essa estratégia permite que os modelos escalem em capacidade e "conhecimento" de forma muito mais sustentável, com uma eficiência computacional muito maior. Embora o modelo como um todo possa ter um número astronômico de parâmetros (a soma de todos os especialistas), o custo para gerar

uma resposta é muito menor, pois apenas uma fração desses parâmetros é utilizada ao mesmo tempo. É poder usar o conhecimento de toda a equipe médica do hospital, mas pagando apenas pela consulta dos especialistas que você efetivamente usou.

Dando um exemplo específico, o modelo *open source Kimi K2* da *Moonshot AI* é um modelo gigantesco, com um total de 1 trilhão de parâmetros e 384 "*experts*", mas apenas 32 bilhões e 8 "*experts*" são ativados de cada vez.[18]

Claro, essa abordagem também tem seus desafios. Um deles é garantir que a rede de roteamento aprenda a distribuir o trabalho de forma equilibrada, evitando que alguns especialistas fiquem sobrecarregados enquanto outros permanecem ociosos. É como garantir que a triagem do hospital não mande todos os pacientes para o mesmo cardiologista, independentemente de seus sintomas.

No fim das contas, a arquitetura *Mixture-of-Experts* se consolidou como um pilar fundamental na evolução dos LLMs, provando que, às vezes, ter uma equipe de especialistas bem coordenada é muito mais inteligente e eficiente do que tentar ter um "sabe-tudo".

E essa busca por eficiência e especialização não parou por aí.

4.8 Modelos de Raciocínio e Híbridos

Outra inovação trazida pela *OpenAI* com o modelo **o1** e que posteriormente virou um padrão no mercado foram os chamados "Modelos de Raciocínio". Calma, isso não quer dizer que os modelos tomaram consciência de si próprios e estão raciocinando como seres humanos (pelo menos ainda não), mas a forma como processam as perguntas passou por uma mudança significativa.

Os modelos de linguagem, até então, recebiam um *prompt* do usuário e começavam a responder (prevendo o próximo *token*)

imediatamente. Um "modelo de raciocínio" adiciona a isso uma etapa intermediária de planejamento e avaliação, geralmente baseada em técnicas como o *Chain-of-Thought* ("Cadeia de Pensamento" em português — veremos mais sobre essa técnica no capítulo sobre Engenharia de *Prompt*) e outras, para fragmentar o problema em subproblemas, analisar e avaliar esses subproblemas, como se estivesse "pensando" nas várias possibilidades antes de responder. É o equivalente daquela pausa que você faz antes de responder à pergunta "*O que você achou do meu novo corte de cabelo?*". O modelo agora faz uma análise interna de riscos e possibilidades antes de se comprometer com uma resposta.

Dando um exemplo bem simplificado, seria como se perguntássemos ao modelo "*Uma camiseta custa R$ 97,00. Maria comprou 2 camisetas e deu uma nota de R$ 200,00. Quanto ela recebeu de troco?*". Os modelos "normais" até então, tentariam prever a resposta (R$ 6,00) diretamente e teriam uma chance mais alta de errar. Já um modelo de raciocínio elaboraria antes uma linha de pensamento no estilo:

"*Ok, vamos resolver isso em etapas: primeiro, vou calcular o custo total das 2 camisetas, que será: 2 x R$ 97,00 = R$ 194,00. Agora, vou calcular o troco com base no pagamento de com uma nota de R$ 200,00: R$ 200,00 - R$ 194,00 = R$ 6,00. Portanto, a Maria recebeu R$ 6,00 de troco.*"

Esse processo de "pensar antes de responder" já era usado indiretamente por desenvolvedores e engenheiros de *prompt* há algum tempo, induzindo os modelos a refletirem com comandos como "*Pense passo a passo*" e "*Explique o seu raciocínio*". A diferença é que os modelos de raciocínio fazem isso de forma nativa, pois já é parte do seu treinamento.

Essa fase intermediária também é realizada pelo próprio modelo apenas gerando o próximo *token*, e cada empresa decide o quanto

dessa fase fica visível para o usuário final. O modelo *o1* original era bastante restritivo nesse quesito, algo que mudou depois do lançamento do modelo *open source DeepSeek-R1* da empresa chinesa *DeepSeek AI*. Modelos *open source* são modelos cuja empresa criadora libera seus pesos de forma aberta, e qualquer pessoa pode fazer o download e rodar localmente, sem ter que pagar o custo por *tokens* da *API* à empresa, como é o caso com modelos privados. O *DeepSeek-R1*, além de ter uma performance excelente, comparável à dos modelos privados, era extremamente transparente quanto ao seu "raciocínio" interno. A aceitação do público foi tão grande que inspirou debates e pressões de mercado e praticamente forçou as outras empresas a adotarem uma postura mais aberta nesse sentido.

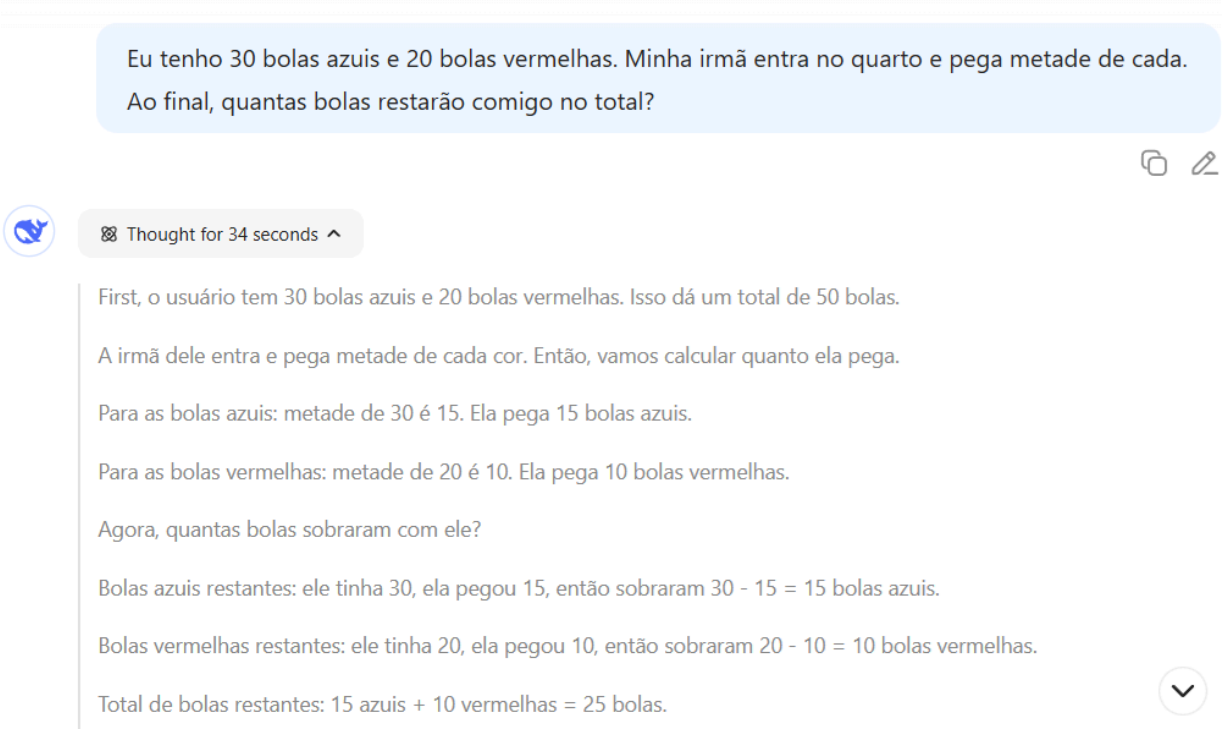

Figura 4.24: Exemplo real de raciocínio do modelo DeepSeek-R1

Além disso, o *DeepSeek-R1* viralizou na época do lançamento, no início de 2025, não só por ser *open source*, transparente com seu raciocínio e ter um desempenho equiparável ao melhor modelo privado da época (o *o1* da *OpenAI*), mas também por supostamente

ter conseguido chegar a esse resultado impressionante com um custo alguns níveis de grandeza menor que o dos concorrentes — supostamente, apenas 5,5 milhões de dólares —, muito devido a restrições impostas pelos Estados Unidos à China sobre a compra das *GPUs* (placas gráficas de alto desempenho usadas para o treinamento e inferência desses modelos) mais avançadas do mercado, o que inclusive causou uma queda acentuada no valor das ações da *NVIDIA*, a principal fornecedora mundial de *GPUs*. Posteriormente, descobriu-se que a história não era bem assim. De qualquer forma, eles foram, de fato, muito inovadores no treinamento do modelo, e subverteram a "receita" popularizada pelo *ChatGPT*, que vimos mais acima.

Você se lembra dos passos que vimos para treinar o primeiro *ChatGPT*? Após o pré-treinamento inicial (Passo 1), ele passou por um intenso Refinamento Supervisionado (SFT) (Passo 2), em que milhares de exemplos de perguntas e respostas ideais, escritas por humanos, foram usados para ensiná-lo a seguir instruções. Só depois ele foi aprimorado com Aprendizado por Reforço (RLHF) (Passos 3 e 4) para alinhar suas respostas ao gosto humano.

A equipe do *DeepSeek* inverteu essa lógica. Partindo também de um modelo base pré-treinado, eles praticamente pularam o massivo Passo 2. Em vez de alimentá-lo com milhares de respostas prontas, eles foram quase direto para o *Aprendizado por Reforço*. A grande inovação foi que o sistema de recompensa não premiava apenas uma resposta final correta, mas sim a demonstração de um processo de raciocínio lógico explícito, através de uma estrutura de "pense e depois responda" (`<think>...</think>`).

E para "dar a partida" nesse processo, eles introduziram um pequeno conjunto de dados de alta qualidade com exemplos de raciocínio escritos por humanos, um "pontapé inicial" (*cold start*) muito mais enxuto e focado do que o SFT tradicional. Como resultado, o modelo aprendeu a raciocinar de forma emergente, desenvolvendo a capacidade de verificar e corrigir a si mesmo. Na prática, enquanto o *ChatGPT* primeiro aprendeu a "imitar" boas

respostas para depois ser refinado, o *DeepSeek-R1* aprendeu primeiro a "pensar" de forma estruturada, o que se provou uma rota muito mais eficiente para desenvolver habilidades de raciocínio complexas. Isso mostra que não existe uma receita de bolo única na hora de treinar esses *LLMs*.

Mas, voltando ao lançamento do *o1* da *OpenAI*, em *benchmarks* de matemática, programação, lógica e ciência aplicada, ele superou não só todos os modelos padrão de até então, como também superou humanos em certos testes específicos. O modelo se mostrou especialmente eficaz em tarefas de raciocínio algorítmico, como problemas de programação, jogos matemáticos e resolução de quebra-cabeças. O sucesso dessa nova arquitetura foi tão grande que virou um padrão no mercado, sendo adotada pelos modelos das famílias *Gemini* do *Google*, *Claude* da *Anthropic*, o já mencionado *DeepSeek-R1*, e até modelos menores, com um tamanho para rodar localmente em dispositivos como *laptops* e celulares, já têm suas versões de raciocínio, tanto os privados como os *open source*.

Outras técnicas também estão sendo pesquisadas e implementadas pelas empresas, como algoritmos que fazem *rollouts* internos (como jogadores de xadrez simulando várias jogadas antes de mover uma peça), estruturas tipo *Tree-of-Thought* ("Árvore de Pensamento" em português, indicando raciocínio com ramificações), e até *Reflexion Loops* ("Laços de Reflexão" em português), todos tentando responder à pergunta: "*como podemos fazer o modelo errar menos, raciocinar melhor e ser mais confiável?*"

Porém, esse avanço não vem sem custos. Esse "raciocínio" continua sendo, no fim das contas, uma geração de *tokens*, assim como qualquer outra resposta. A diferença é que agora, antes mesmo de entregar algo ao usuário, o modelo pode gerar até milhares de *tokens* nessa fase intermediária para planejar, explorar hipóteses e verificar inconsistências, tudo isso durante a própria inferência (a fase de geração da resposta). Como consequência, as *GPUs*

precisam trabalhar por mais tempo para concluir cada resposta, já que a quantidade de processamento por requisição aumentou significativamente. Isso tem implicações práticas importantes: os custos operacionais sobem (seja para empresas que hospedam os modelos ou usuários que pagam por uso), a latência pode crescer em contextos sensíveis ao tempo, e a escalabilidade precisa ser repensada.

Porém, nem todas as perguntas precisam de um grande raciocínio para serem respondidas. Por exemplo, traduzir ou criar resumos de textos são tarefas que modelos sem raciocínio faziam e fazem extremamente bem. Com isso, e para mitigar esse custo computacional crescente, as empresas começaram a treinar **modelos híbridos**, que aprendem a avaliar a complexidade da consulta de forma dinâmica durante a inferência para saber se devem escolher entre um modo "rápido" (resposta direta) ou de "raciocínio" para gerar a resposta, otimizando desempenho, custo e latência. Se a pergunta for simples (*"Qual é a capital da França?"*), o modelo responde diretamente; se envolver múltiplas etapas ("*Como fazer um planejamento financeiro para um autônomo com renda variável?"*), ele aciona seu modo de "raciocínio".

O modelo *Claude 3.7 Sonnet* da *Anthropic* foi o primeiro modelo híbrido comercial a ser lançado, seguido pelos modelos da família *Gemini 2.5* do *Google*.

4.9 Uso de ferramentas externas

Outra melhoria trazida pelos modelos mais recentes foi a de usar ferramentas externas. Modelos de linguagem têm uma dificuldade inerente para realizar certas tarefas, como operações matemáticas, porque eles lidam apenas com texto. É muito fácil prever que 2 + 2 = 4, pois essa é uma informação que ele viu muito em textos da internet durante o seu treinamento. Mas e para uma conta que não

aparece com tanta frequência, como 3945745 x 853023? Ele teria grandes chances de se perder.

Por isso, os modelos de ponta hoje em dia não tentam fazer esse tipo de cálculo internamente. Eles possuem acesso a ferramentas e são treinados para saber quando usá-los, algo como: "*Ok, isso é uma multiplicação. Vou acionar a ferramenta de calculadora com os valores 3945745 e 853023*". Isso mostra que até um cérebro digital com trilhões de parâmetros tem a humildade de admitir que, para algumas coisas, uma simples calculadora é a melhor opção. É um alívio saber que, assim como nós, ele também não quer fazer contas "de cabeça".

Tecnicamente, esse processo é conhecido como "*Function Calling*" ("Chamada de Função" em português). O modelo de linguagem não executa a calculadora diretamente. Em vez disso, durante o seu treinamento, ele aprende a reconhecer uma tarefa que necessita de uma ferramenta e a formatar sua resposta em um código estruturado (geralmente em formato *JSON*), especificando qual ferramenta chamar e quais argumentos (os números da multiplicação, nesse caso específico) passar para ela. Um sistema externo, então, lê esse código, executa a ferramenta real (a calculadora) com os argumentos fornecidos e devolve o resultado para o modelo de linguagem. Com o resultado em mãos, o modelo continua a geração de sua resposta final em linguagem natural.

Algumas das ferramentas mais usadas hoje em dia pelos modelos são a calculadora, um interpretador de código em linguagem de programação *Python* (que permite acesso a bibliotecas de análise de dados, como a *Pandas*, e assim, lidar com valores em tabelas e planilhas) e uma *API* de busca de informações na internet, que permite que os modelos tenham acesso a informações atuais, que aconteceram muito depois da data do seu treinamento. Todos os *LLMs* mais conhecidos já possuem esse tipo de acesso a ferramentas.

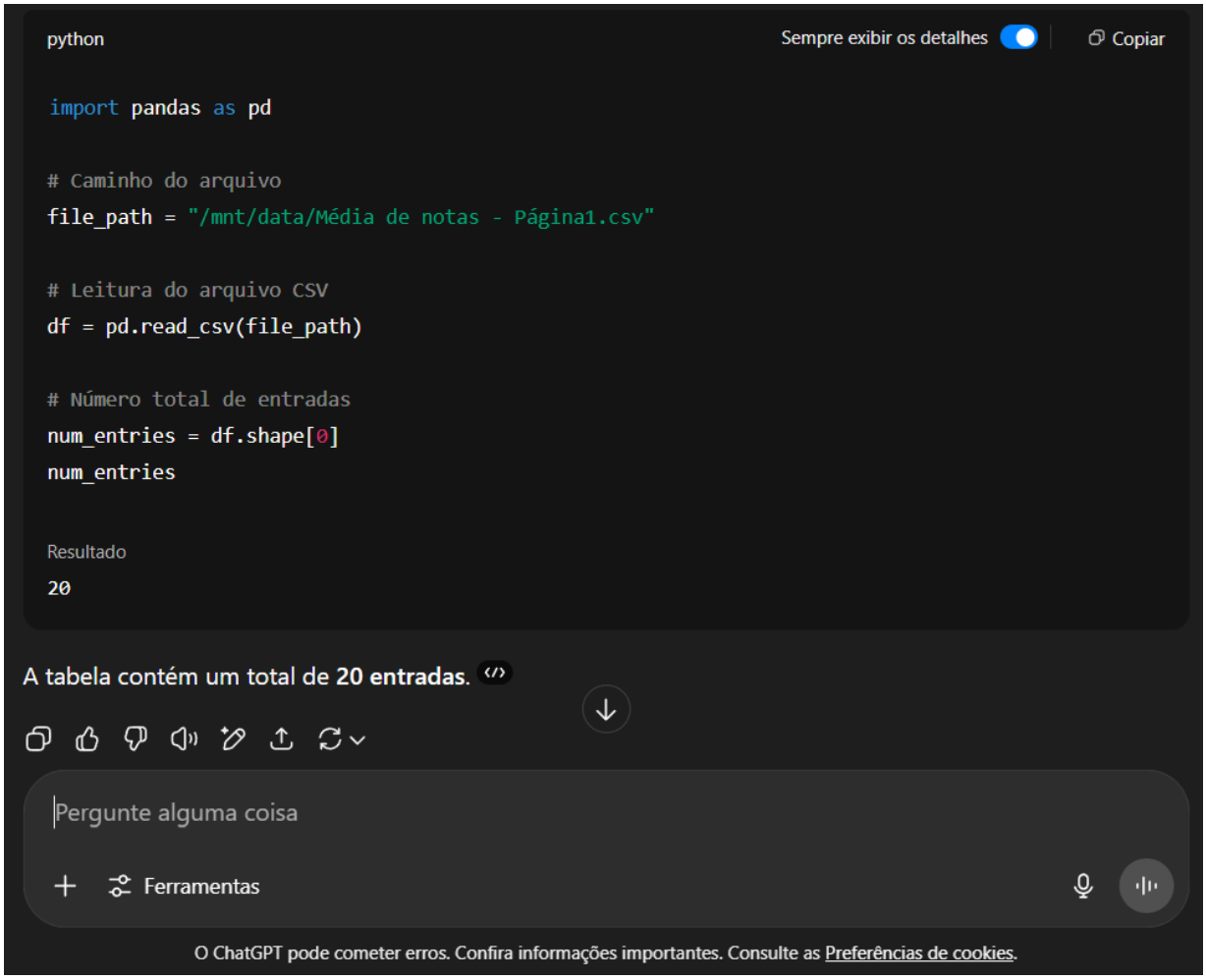

Figura 4.25: Exemplo do ChatGPT usando o interpretador de código em Python e a biblioteca Pandas para analisar uma planilha em formato CSV

Com isso, os modelos de linguagem começam cada vez mais a se tornar na verdade **agentes** (uma palavra importantíssima, que será abordada em mais detalhes em breve), treinados para "agir" e tomar decisões sobre como e com qual ferramenta tentar encontrar a solução para determinada tarefa, em vez de apenas tentar prever essa solução diretamente através de texto.

Outra ferramenta também muito útil para o usuário final é o que a *Anthropic* chama de "*Artifacts*", e tanto a *OpenAI* quanto o *Google* chamam de *Canvas*. Pense nisso como uma bancada de trabalho ou um quadro branco interativo que fica ao lado da sua conversa principal com a IA. A função desse espaço é abrigar os "artefatos"

ou resultados úteis gerados pelo modelo, como blocos de código, designs de interface, documentos de texto, gráficos ou fluxogramas.

Por exemplo, se você pede ao modelo para criar a página inicial de site usando código (HTML e CSS), ele não só escreve o código, mas também o renderiza em tempo real nesta área de trabalho virtual. Você pode então interagir com esse componente, pedir ao modelo para fazer alterações no código e ver o resultado visual se atualizar instantaneamente, sem precisar copiar, colar e testar em outro ambiente.

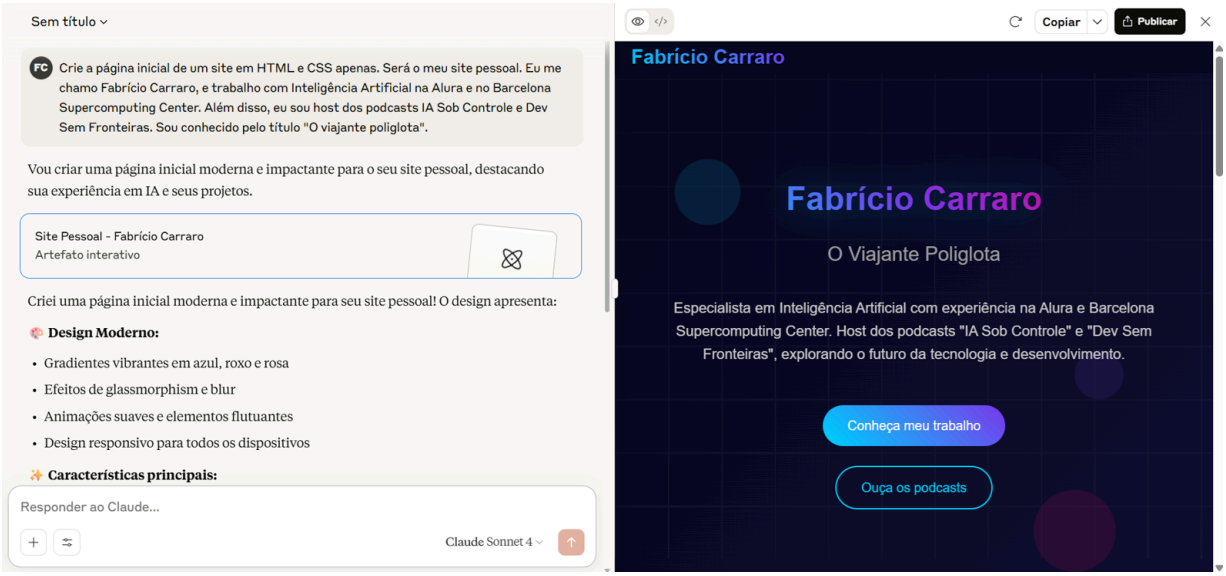

Figura 4.26: Exemplo de um site sendo exibido nos Artifacts do Claude

4.10 Deep Research, a Busca Profunda

Dado que os modelos já conseguiam "raciocinar" por mais tempo (em vez de começar a responder imediatamente) e também já tinham acesso a ferramentas externas, tornando-os mais próximos de agentes, que podiam tomar decisões, a *OpenAI* apresentou mais

uma funcionalidade extremamente útil (e posteriormente adotada por outros produtos): a *Deep Research* ("Busca Profunda", em português).

Imagine que você peça ao modelo: "*Analise o impacto econômico para o agronegócio brasileiro do novo acordo comercial entre o Mercosul e o Japão, anunciado no mês passado, citando as projeções de diferentes consultorias* ".

Uma pesquisa simples pela ferramenta de busca do modelo se limitaria a fazer uma pesquisa, geralmente rápida, nos principais sites retornados sobre aquele assunto, e gerar uma resposta a partir do conhecimento lido. Não seria suficiente para algo tão complexo. Já a função de *Deep Research*, ao receber esse prompt, inicia um processo de várias fases, que se assemelha ao trabalho de um analista de pesquisa humano.

Primeiro, o modelo de raciocínio interno decompõe a pergunta principal em uma série de subperguntas e tarefas de pesquisa, como: "*Existe um novo acordo comercial Mercosul-Japão?*", "*Quando foi anunciado?*", "*Quais são os principais pontos do acordo para o agronegócio?*", "*Quais consultorias publicaram análises sobre isso?*", "*Pesquisar: 'projeções econômicas acordo Mercosul-Japão agronegócio'.*", "*Pesquisar: 'análise* `[Nome da Consultoria 1]` *acordo Mercosul-Japão'.*" etc.

O modelo, então, executa essas buscas de forma autônoma na internet. Ele não acessa só a primeira página de resultados, mas pode navegar por dezenas ou até centenas de links, ler artigos de notícias, relatórios em PDF, postagens em blogs de especialistas e documentos oficiais. Ele funciona como um navegador inteligente, extraindo os trechos de texto mais relevantes de cada fonte.

Tendo coletado os dados, as organiza as informações, compara dados de diferentes fontes, identifica pontos de consenso e de divergência entre as análises, extrai números e estatísticas relevantes e estrutura tudo isso em uma base de conhecimento

temporária, criada especificamente para responder à sua pergunta. Se uma fonte diz que o impacto será de 5 bilhões de dólares e outra diz 8 bilhões, o modelo anota essa diferença.

Por fim, o modelo utiliza sua capacidade de geração de linguagem natural para redigir uma resposta coesa e bem estruturada. O resultado não é uma lista de links, mas um relatório completo de muitas páginas, seções, resumos, citações e referências às fontes originais, permitindo que o usuário verifique a procedência de cada informação.

É como ter uma equipe de pesquisadores trabalhando para você: eles recebem a tarefa, vasculham a biblioteca e a internet, compilam os resultados, cruzam as informações e lhe entregam um relatório consolidado para sua análise final, em um processo que pode levar de 10 a 30 minutos.

Vale lembrar que, apesar de menor, o risco de alucinação não desaparece. O modelo pode interpretar mal uma fonte ou combinar informações de maneiras que criem uma conclusão factualmente incorreta, ainda que plausível e bem redigida. A verificação das referências continua sendo fundamental para mitigar esse risco.

Além do *ChatGPT*, modelos como o *Gemini* do *Google* e o *Grok* da *xAI* já incorporam essa funcionalidade, cada um com sua própria abordagem e nível de sofisticação.

Com esses modelos ficando cada vez mais poderosos, chegamos a um novo ponto da sua evolução: transformá-los em **agentes**, capazes não só de analisar dados e trazer resultados, mas de tomar decisões, voltar atrás e corrigir uma decisão equivocada, a fim de chegar ao resultado desejado de forma totalmente (ou quase) autônoma. Esse será o tema do próximo capítulo.

Referências bibliográficas:

[1] HU, Krystal. ChatGPT sets record for fastest-growing user base - analyst note. *Reuters*, 2023. Disponível em: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/

[2] HUGGING FACE. Summary of the tokenizers. c2023. Disponível em: https://huggingface.co/docs/transformers/tokenizer_summary

[3] AHIA, Orevaoghene; KUMAR, Sachin; GONEN, Hila; KASAI, Jungo; MORTENSEN, David R.; SMITH, Noah A.; TSVETKOV, Yulia. Do All Languages Cost the Same? Tokenization in the Era of Commercial Language Models. *Research Gate*, 2023. Disponível em: https://www.researchgate.net/publication/370981335_Do_All_Languages_Cost_the_Same_Tokenization_in_the_Era_of_Commercial_Language_Models

[4] BROWN, Tom B. *et al*. Language Models are Few-Shot Learners. *arXiv*, 2020. Disponível em: https://arxiv.org/abs/2005.14165

[5] OUYANG, Long *et al*. Training language models to

follow instructions with human feedback. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2203.02155

[6] VANIAN, Jonathan; LESWING, Kif. ChatGPT and generative AI are booming, but the costs can be extraordinary. *CNBC*, 2023. Disponível em: https://www.cnbc.com/2023/03/13/chatgpt-and-generative-ai-are-booming-but-at-a-very-expensive-price.html

[7] OPENAI. *Introducing ChatGPT*. 2022. Disponível em: https://openai.com/blog/chatgpt

[8] SCHULMAN, John; WOLSKI, Filip; DHARIWAL, Prafulla; RADFORD, Alec; KLIMOV, Oleg. Proximal Policy Optimization Algorithms. *arXiv*, 2017. Disponível em: https://arxiv.org/abs/1707.06347

[9] ROTH, Emma. Microsoft spent hundreds of millions of dollars on a ChatGPT supercomputer. *The Verge*, 2023. Disponível em: https://www.theverge.com/2023/3/13/23637675/microsoft-chatgpt-bing-millions-dollars-supercomputer-openai

[10] WEISE, Karen; METZ, Cade. When A.I. Chatbots

Hallucinate. *The New York Times*, 2023. Disponível em: https://www.nytimes.com/2023/05/01/business/ai-chatbots-hallucination.html

[11] OPENAI. *GPT-4*. 2023. Disponível em: https://openai.com/research/gpt-4

[12] SCHREINER, Maximilian. GPT-4 architecture, datasets, costs and more leaked. *The Decoder*, 2023. Disponível em: https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

[13] KNIGHT, Will. OpenAI's CEO Says the Age of Giant AI Models Is Already Over. *Wired*, 2023. Disponível em: https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/

[14] KARHADE, Mandar. GPT-4: 8 Models in One; The Secret is Out. *Towards AI*, 2023. Disponível em: https://pub.towardsai.net/gpt-4-8-models-in-one-the-secret-is-out-e3d16fd1eee0

[15] MEHDI, Yusuf. Confirmed: the new Bing runs on OpenAI's GPT-4. *Microsoft Bing Blogs*, 2023. Disponível em: https://blogs.bing.com/search/march_2023/Confirme

d-the-new-Bing-runs-on-OpenAI%E2%80%99s-GPT-4

[16] WIGGERS, Kyle. ChatGPT users have generated over 700M images since last week, OpenAI says. *TechCrunch*, 2025. Disponível em: https://techcrunch.com/2025/04/03/chatgpt-users-have-generated-over-700m-images-since-last-week-openai-says/

[17] Gemini 2.5 Pro Model Card. *Google*, 2025. Disponível em: https://storage.googleapis.com/model-cards/documents/gemini-2.5-pro.pdf

[18] Kimi K2: Open Agentic Intelligence. *Moonshot AI*, 2025. Disponível em: https://moonshotai.github.io/Kimi-K2/

Capítulo 5
Agentes de IA

Nos capítulos anteriores, vimos como os modelos de linguagem funcionam, como foram treinados e como se transformaram em ferramentas cada vez mais poderosas, capazes de raciocinar, acessar ferramentas e buscar na internet. Mas... e se eles pudessem ir além? E se, em vez de apenas reagirem a comandos, esses modelos pudessem tomar a iniciativa, planejar múltiplas etapas, corrigir erros sozinhos, ou até coordenar com outros modelos?

Você deve ter notado que eu usei a palavra "agente" algumas vezes. E não foi por acaso. Prepare-se: este capítulo marca uma virada conceitual. Bem-vindo(a) ao mundo dos **Agentes de IA**, o ponto em que os modelos de IA deixam de ser apenas "enciclopédias generativas" e passam a agir como assistentes autônomos, capazes de interagir com o ambiente, executar tarefas complexas e atingir objetivos específicos com (quase) nenhuma intervenção humana.

Neste capítulo, vamos mergulhar de cabeça nesse que é um dos campos mais excitantes e de rápida evolução da Inteligência Artificial. Vamos entender o que são esses agentes, como funcionam, quais são as tecnologias que os impulsionam e por que eles prometem ser a próxima grande revolução.

5.1 O que é um Agente de IA?

Vamos, mais uma vez, tirar o elefante da sala. Um agente de IA é um sistema que percebe seu ambiente, toma decisões de forma autônoma e executa ações para atingir objetivos específicos. Simples, não? A beleza (e a complexidade) está no que cada uma dessas partes significa no contexto dos *LLMs* que estudamos.

No contexto de IA generativa com modelos de linguagem, quando falamos de agentes, normalmente nos referimos a sistemas que usam um LLM como seu "cérebro", mas acoplado a outros componentes que lhe dão memória, ferramentas para agir e, muitas vezes, diretrizes de planejamento. Uma definição citada pela própria NVIDIA diz que um agente é "*um sistema que utiliza um LLM para raciocinar sobre um problema, criar um plano de solução e executar esse plano com a ajuda de um conjunto de ferramentas*"[1] .

Imagine que você contratou um estagiário genial (o nosso *LLM*). Inicialmente, você pede a ele: "*Escreva um resumo sobre a exportação de café a partir do Brasil*". Ele usa todo o conhecimento que tem, buscas na internet e escreve um relatório completíssimo. Agora, imagine que você diga: "*Planeje minha viagem de férias para o litoral da Bahia. O orçamento total é de R$ 5.000,00. Encontre os voos mais baratos para a primeira semana de dezembro e efetue a compra do mais conveniente. Sugira três hotéis com boas avaliações e piscina, e reserve aquele que tenha o melhor custo-benefício*". O estagiário (*LLM*) comum travaria. O estagiário que é um Agente de IA, no entanto, arregaça as mangas e começa a agir. Ele abre o navegador, busca por voos, filtra os resultados, acessa sites de reserva de hotéis, lê as avaliações e, no final, apresenta para você o plano completo, quem sabe até já usando o seu cartão de crédito para fazer as reservas (caso você o tenha autorizado para tal).

Essa é a diferença fundamental. Os Agentes de IA marcam a transição de um modelo que *sabe* para um modelo que *faz*. A palavra-chave aqui é **ação**.

É aquele estagiário que não apenas traz o café, mas que planta o grão, colhe, torra, mói e ainda lembra que você tem uma reunião em cinco minutos. O nível de proatividade é outro patamar.

A ascensão dos LLMs foi um catalisador para o desenvolvimento de agentes de IA mais sofisticados. Antes, a construção de agentes autônomos era um desafio complexo, exigindo programação explícita para cada cenário. Com os *LLMs*, os agentes ganham a capacidade

de raciocinar em linguagem natural, planejar tarefas complexas, aprender com a experiência e até mesmo se adaptar a situações imprevistas. Eles podem "pensar" sobre o problema, "decidir" a melhor abordagem e "executar" a solução, tornando-se verdadeiros solucionadores de problemas digitais.

5.2 Como funcionam Agentes de IA

Ok, entendemos o "quê", mas como exatamente um agente "pensa" e "age"? O conceito de agentes é sustentado por três pilares fundamentais que definem o seu comportamento:

- **Autonomia**: a capacidade de operar com intervenção humana mínima ou nenhuma após receber um objetivo. Uma vez acionado, o agente pode perceber seu ambiente, raciocinar sobre os dados e executar ações de forma independente para progredir em direção à sua meta.
- **Orientação a objetivos (*Task-Specificity*)**: os agentes são construídos com um propósito. Eles são otimizados para executar operações dentro de um domínio específico, como filtrar e-mails, consultar um banco de dados ou coordenar um calendário. Essa especialização garante eficiência e precisão.
- **Reatividade e adaptação**: um agente não opera no vácuo. Ele deve ser capaz de perceber estímulos em tempo real (como uma nova solicitação do usuário, uma resposta de uma API ou uma mudança de estado em um sistema) e adaptar seu plano e suas ações de acordo.

Para que um agente possa planejar e agir de forma autônoma, ele precisa de uma arquitetura interna que combine raciocínio, memória e a capacidade de interagir com o mundo exterior. Essa arquitetura não é apenas uma pilha de tecnologias, mas uma tentativa de emular o modo como nós, seres humanos, planejamos e realizamos

ações. Em termos simples, os seus componentes principais são o cérebro, a memória, o planejamento e as ferramentas.

- **Cérebro**: o cérebro de todo agente será um *LLM*, que não só vai ser o responsável por gerar texto, mas será motor de raciocínio do agente. Ele receberá um objetivo, o analisará, considerará o contexto disponível (da memória e das observações recentes) e raciocinará sobre a melhor sequência de passos a seguir. É o *LLM* que decide se precisa de mais informações, qual ferramenta usar para obtê-las e como interpretar os resultados para planejar a próxima ação.
- **Memória**: a memória é o que eleva um agente de uma ferramenta sem estado, que trata cada interação como a primeira, a um parceiro contextualizado, capaz de aprender e se adaptar ao longo do tempo. A arquitetura de memória de um agente é frequentemente dividida em duas categorias principais: memória de curto e longo prazo. Vale dizer que nem todos os agentes vão conter essas memórias, alguns podem ter apenas uma delas, ou até nenhuma.
    - **Memória de curto prazo**: seria como um "bloco de notas" da conversa atual, que armazena o objetivo original, as ações já tomadas e as observações coletadas. Tecnicamente, isso é muitas vezes implementado através da janela de contexto do LLM ou de um "buffer" de mensagens que armazena o histórico recente da interação. Uma vez que a tarefa é concluída, a informação é tipicamente descartada.
    - **Memória de longo prazo**: é um repositório persistente de informações, geralmente implementado com bancos de dados vetoriais, onde as informações são armazenadas como *embeddings* e podem ser recuperadas semanticamente, permitindo que o agente "aprenda" com experiências passadas ou consulte uma base de conhecimento massiva (como todos os documentos internos da sua empresa).

- **Planejamento**: antes de agir, um agente eficaz deve planejar. O módulo de planejamento é responsável por decompor um objetivo grande e ambíguo em uma série de subtarefas menores, concretas e gerenciáveis. Este processo de decomposição cria um roteiro ou um plano de ação que guiará a execução do agente.
- **Ferramentas (Tools)**: um *LLM*, por si só, está confinado ao seu próprio conhecimento e só pode produzir texto. As ferramentas são a ponte que conectam o raciocínio do modelo de linguagem ao mundo real, permitindo que ele execute ações concretas. O mecanismo que torna isso possível é comumente chamado de *Function Calling* ou *Tool Use*.

Além disso, para que um agente execute tarefas complexas de forma confiável, ele precisa de mais do que apenas componentes; ele precisa de padrões de raciocínio e arquiteturas que estruturem seu comportamento. Um dos padrões mais influentes e fundamentais é o *ReAct*[2] , um paradigma que estrutura o processo de tomada de decisão de um agente.

O fluxo que ele propõe intercala três fases principais:

1. **Pensamento**: "*Preciso encontrar informação X. Para isso, preciso chamar a ferramenta Y.*";
2. **Ação**: Chamada da ferramenta "Y";
3. **Observação**: "*Encontrei tal resultado contendo X*".

E assim seguir, iterativamente, refinando o plano e avançando nas tarefas. Algo como:

Pensamento → Ação → Observação → Pensamento → Ação...

É o mesmo ciclo que usamos para montar um móvel das *Casas Bahia*: **Pensamento** (*'Ok, este parafuso parece ir aqui.'*), **Ação** (*tentar encaixar o parafuso*), **Observação** (*'Não, definitivamente*

não vai aqui. Sobrou uma peça.'). A diferença é que o agente não se frustra nem joga o manual pela janela.

Vamos acompanhar esse ciclo com um exemplo prático. Imagine que você dê a um agente o seguinte objetivo: "*Analise as vendas do último trimestre a partir do arquivo 'vendas.csv', compare com os dados públicos da empresa concorrente 'var X' e me envie um resumo por e-mail com os 3 produtos mais vendidos*". O agente poderia seguir os seguintes passos:

Passo 1 - Entender o arquivo e os dados:

1. **Pensamento**: "*Ok, o usuário quer que eu analise um arquivo chamado 'vendas.csv' e depois compare os seus dados com os dados públicos da empresa concorrente 'var X'. A primeira coisa que preciso fazer é entender o que há dentro desse arquivo. Quais são as colunas? Como os dados estão estruturados? A melhor ferramenta para isso é o meu interpretador de código, provavelmente usando a biblioteca Pandas em Python para ler o arquivo e mostrar as primeiras linhas.*"
2. **Ação**: Usar a ferramenta Interpretador de Código Python.

```
import pandas as pd
df = pd.read_csv('vendas.csv')
print(df.head())
```

3. **Observação**: O agente recebe a saída do código, que mostra as primeiras 5 linhas do arquivo CSV. Ele vê colunas como '*ID_Produto*', '*Nome_Produto*', '*Quantidade_Vendida*', '*Preco_Unitario*', '*Data_Venda*'.

Passo 2 - Calcular os produtos mais vendidos:

1. **Pensamento**: "*Certo, agora eu entendo a estrutura. A coluna 'Nome_Produto' contém o nome dos produtos e a 'Quantidade_Vendida' contém a quantidade. Para encontrar os 3 mais vendidos, preciso agrupar os dados por 'Nome_Produto', somar as 'Quantidade_Vendida' de cada um e depois ordenar o resultado em ordem decrescente para pegar os 3 primeiros.*"
2. **Ação**: Usar a ferramenta Interpretador de Código Python.

```
produtos_mais_vendidos = df.groupby('Nome_Produto')
['Quantidade_Vendida'].sum().sort_values(ascending=False).head(3)


print(produtos_mais_vendidos)
```

3. **Observação**: O agente recebe a saída do código, que é uma lista dos 3 produtos mais vendidos e suas quantidades totais. Por exemplo: *{'Produto A': 540, 'Produto C': 480, 'Produto B': 455}*.

Passo 3 - Buscar os dados públicos da concorrente:

1. **Pensamento**: "*Agora, vou precisar acessar os dados da empresa concorrente 'var X' para compará-los. Como são dados públicos, acredito que estejam disponíveis na internet. Para isso, vou usar a minha ferramenta de buscas na internet.*"
2. **Ação**: Usar ferramenta de busca na internet. Buscar "*dados de vendas da empresa var X para o último trimestre*".
3. **Observação**: O agente recebe como retorno vários sites com dados que podem ser os desejados.

Passo 4 - Vasculhar o conteúdo dos sites para extrair a informação:

1. **Pensamento**: "*Já tenho o conteúdo dos sites que a minha ferramenta de busca retornou. Agora, vou analisar todos para extrair os dados desejados.*"
2. **Ação**: Usar o próprio *LLM* (cérebro) para analisar o conteúdo retornado e encontrar os dados desejados. Texto gerado internamente: "*As vendas dos produtos A, B e C pela empresa 'var X' no último trimestre foram* {'Produto C': 900, 'Produto B': 750, 'Produto A': 600}._"
3. **Observação**: O agente recebe como retorno vários sites com dados que podem ser os desejados.

Passo 5 - Elaborar o e-mail:

1. **Pensamento**: "*Excelente, já tenho a informação principal. Agora preciso criar o corpo do e-mail. O texto deve ser claro e direto, apresentando o resumo solicitado. Vou formatar a lista que obtive em um texto amigável.*"
2. **Ação**: Usar o próprio *LLM* (cérebro) para gerar o texto. Texto gerado internamente: "*Olá, segue o resumo de vendas do último trimestre. Os 3 produtos mais vendidos foram: 1. Produto A (540 unidades) 2. Produto C (480 unidades) 3. Produto B (455 unidades). Os dados públicos da empresa 'var X' demonstram que seus produtos mais vendidos foram 1. Produto C (900 unidades, 'Produto B': 750, 'Produto A': 600}. Com isso, podemos ver que as vendas deles são maiores para todos os produtos em questão. Atenciosamente, Seu Agente de IA.*"
3. **Observação**: O agente recebe o texto gerado, a ser enviado por e-mail.

Passo 6 - Enviar o e-mail:

1. **Pensamento**: "*O texto do e-mail está pronto. O destinatário é (e-mail do usuário que fez a solicitação) e o assunto pode ser 'Resumo de Vendas do Último Trimestre'. Agora, preciso usar a ferramenta de envio de e-mail do Gmail para enviá-lo.*"
2. **Ação**: Usar a ferramenta "Enviar E-mail" do Gmail. Parâmetros da ferramenta: "destinatario": *fabricio.carraro@alura.com.br*; "assunto": "*Resumo de Vendas do Último Trimestre*"; "corpo": (*o corpo do e-mail gerado no passo 5*).
3. **Observação**: A ferramenta de e-mail do Gmail retorna uma mensagem de sucesso, como *{'status': 'sucesso', 'mensagem': 'E-mail enviado com sucesso.'}*.

Passo 7- Finalizar a tarefa:

1. **Pensamento**: "*O e-mail foi enviado e a análise foi concluída. Todas as partes do objetivo inicial foram atendidas. A tarefa está finalizada.*"
2. **Ação**: Enviar uma resposta final ao usuário. Resposta final: "*Pronto! A análise de vendas foi concluída e um resumo com os 3 produtos mais vendidos e sua comparação com os dados públicos da empresa concorrente 'var X' foi enviado para o seu e-mail.*"
3. **Observação**: A resposta final foi enviada ao usuário. O processo está terminado.

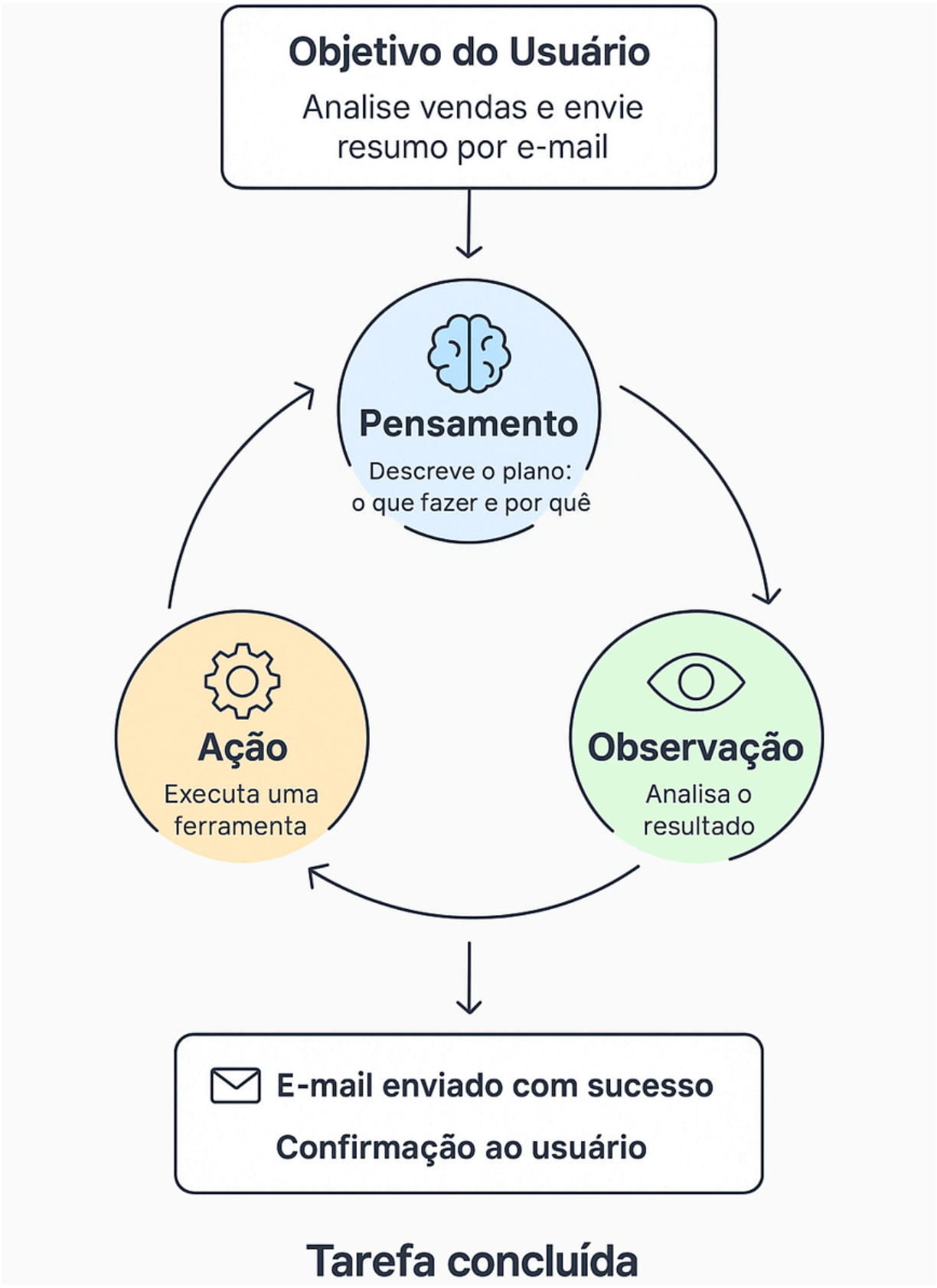
Objetivo do Usuário
Analise vendas e envie resumo por e-mail
Pensamento
Descreve o plano: o que fazer e por quê
Ação
Executa uma ferramenta
Observação
Analisa o resultado
E-mail enviado com sucesso
Confirmação ao usuário
Tarefa concluída

Figura 5.1: Fluxograma do sistema de agentes com base em ReAct

E tudo isso, claro, acontece em segundos, sem pausas para o café ou reclamações sobre a complexidade da planilha. A eficiência de um agente pode ser um tanto quanto humilhante para nós, meros mortais.

5.3 Sistemas multiagentes

Até agora, exploramos como um único agente de IA pode raciocinar, planejar e usar ferramentas para executar tarefas complexas. Porém, assim como no mundo humano, algumas tarefas são grandes ou multifacetadas demais para um único indivíduo, não importa o quão talentoso ele seja. É aqui que entramos na próxima fronteira da IA autônoma: os sistemas multiagentes.

A ideia é simples: em vez de construir um único agente com um *prompt* de sistema gigantesco e acesso a diversas ferramentas diferentes, por que não criar uma equipe de agentes especialistas que colaboram para atingir um objetivo comum?

Pense em como uma agência de publicidade funciona. Você não tem uma única pessoa que cria a estratégia, escreve o texto do anúncio, desenha o visual, produz o vídeo e negocia a veiculação na mídia. Em vez disso, você tem um estrategista, um redator (*copywriter*), um diretor de arte, um produtor e um analista de mídia. Cada um é mestre em sua área, e o resultado final é fruto da sinergia e colaboração entre eles, geralmente orquestrada por um gerente de projetos.

A grande vantagem é que, em uma equipe de agentes, não há disputas de ego, reuniões que poderiam ter sido um e-mail ou discussões sobre quem deixou a louça suja na pia da copa. A colaboração é puramente focada na tarefa, o sonho de qualquer

gerente de projetos. Um sistema multiagentes aplica exatamente essa lógica ao mundo da IA.

Algumas vantagens dessa abordagem:

- **Especialização**: cada agente pode ser afinado (com prompt, ferramentas, contexto e até modelo diferente) para executar melhor um determinado tipo de tarefa. Um "agente programador" pode ser bem diferente de um "agente jurídico".
- **Modularidade**: sistemas multiagentes são mais fáceis de escalar e manter. Se um agente quebra ou precisa ser atualizado, o resto do sistema pode continuar operando normalmente.
- **Paralelismo**: múltiplos agentes podem executar tarefas em paralelo (ou de forma assíncrona), acelerando pipelines mais complexos.
- **Simulação de organizações humanas**: ao distribuir papéis como em uma empresa (gerente, analista, redator etc.), conseguimos simular fluxos de trabalho reais, com dinâmicas de deliberação, consenso e divisão de tarefas.
- **Resiliência e redundância**: um agente pode revisar o trabalho de outro, apontar falhas ou tomar decisões alternativas. Essa diversidade de "pontos de vista" pode reduzir alucinações ou erros grosseiros.

Nesse contexto, um dos grandes desafios dos sistemas multiagentes é a coordenação entre eles, ou seja, como garantir que os agentes não tenham um conflito, que não haja sobreposição desnecessária ou que as informações fluam corretamente entre eles?

Existem alguns padrões clássicos de coordenação, que também estão sendo adaptados a esses sistemas:

- **Planejador + Executores (*Planner-Executor*)**: um agente recebe o objetivo geral, o divide em subtarefas e as delega para outros agentes especializados que as executam. Esse padrão é usado, por exemplo, no *AutoGen* da *Microsoft* e em configurações do *framework CrewAI* (com o nome "*Coordinator*").
- **Conversação simulada (*Role-play Chat*)**: agentes se comunicam como se fossem pessoas em um chat, trocando mensagens com argumentos, sugestões ou propostas. Isso é explorado em frameworks como *Camel* e *MetaGPT*, onde um agente "cliente" discute com um "engenheiro" e um "planejador de projeto" para tomar decisões.
- **Agente Orquestrador (*Orchestrator Agent*)**: um agente tem a visão geral do sistema e atua como intermediador das interações e fluxos entre os demais. Ele recebe as respostas dos agentes especialistas e toma decisões sobre o que fazer em seguida.
- **Colaboração Distribuída (*Peer-to-Peer*)**: todos os agentes são "iguais" em termos de hierarquia e colaboram em pé de igualdade, muitas vezes votando na melhor opção a ser seguida, negociando ou trocando informações livremente. Ainda raros, esse modelo lembra simulações sociais e pode ser útil em tarefas de consenso.

Independentemente do padrão, os sistemas multiagentes geralmente exigem mecanismos de memória compartilhada, controle de tarefas, contexto individual por agente, e algum sistema de log ou *tracing* (como o *LangSmith*, no caso do framework *LangGraph*) para que possamos acompanhar o que cada agente está fazendo.

5.4 Níveis de agência

A transição para sistemas com agentes de IA não é um interruptor binário, mas sim um espectro de autonomia. Para empresas e líderes de tecnologia, entender essa progressão é vital para planejar a adoção e gerenciar os riscos. Uma das taxonomias possíveis oferece um framework de quatro níveis de maturidade para essa jornada:

- **Nível 1 – Cadeia (*Chain*)**: representa a automação de processos tradicional. Tanto as ações quanto a sequência em que são executadas são predefinidas e rígidas. Um exemplo clássico é um script que extrai dados de faturas em PDF e os insere em um sistema de contabilidade.
- **Nível 2 – Fluxo de trabalho (*Workflow*)**: nesse nível, as ações ainda são predefinidas (como "enviar e-mail" ou "consultar base de conhecimento"), mas a sequência é determinada dinamicamente, muitas vezes por um LLM. É aqui que a maioria das aplicações de "IA generativa" se encontra hoje, como os pipelines de *RAG (Retrieval-Augmented Generation*).
- **Nível 3 – Parcialmente autônomo**: aqui, o verdadeiro comportamento agêntico começa a emergir. Dado um objetivo claro e um conjunto de ferramentas específicas do domínio, o agente pode planejar, executar e ajustar uma sequência de ações com supervisão humana mínima. Um exemplo seria um agente de suporte ao cliente que, ao receber um ticket, pode consultar o CRM, verificar o status do pedido no sistema de logística e se comunicar com o cliente para resolver o problema de ponta a ponta.
- **Nível 4 – Totalmente autônomo**: este é o ápice da agência. O sistema opera com pouca ou nenhuma supervisão humana, mesmo em múltiplos domínios. Ele pode definir proativamente seus próprios subobjetivos, adaptar-se a resultados inesperados e até mesmo selecionar ou criar suas próprias ferramentas para cumprir uma missão de alto nível. Um agente de pesquisa estratégica que descobre, resume e sintetiza informações de

forma independente para produzir um relatório de mercado é um exemplo.

Atualmente, a maioria das aplicações empresariais está nos níveis 1 e 2, e estamos vendo a migração para o nível 3 em domínios bem definidos. Podemos ver esses níveis como um roteiro estratégico também. Os desafios de confiabilidade, segurança e custo tornam um salto direto para o Nível 4 imprudente para a maioria das organizações.

Ao começar com a automação de fluxos de trabalho (Nível 2), as empresas podem gerar valor imediato enquanto desenvolvem a infraestrutura, a governança e os *guardrails* ("barreiras de proteção" em português) necessários para gerenciar sistemas mais autônomos. A transição para o Nível 3 representa a verdadeira "agentificação", mas dentro de um escopo controlado, permitindo que a organização construa confiança na tecnologia. Este caminho incremental alinha o investimento tecnológico com a maturidade organizacional, mitigando os riscos de implantar sistemas autônomos em ambientes de produção críticos e permitindo que os líderes orquestrem essa evolução de forma responsável.

Auto-otimização

Mas e se o nosso "estagiário genial" pudesse, além de tudo, aprender com os próprios erros e otimizar seu jeito de trabalhar sem que você precise dizer a ele como?

Essa é a fronteira da pesquisa em agentes de IA: a **auto-otimização**[3] . A ideia é que os agentes podem ser projetados para se aprimorarem de forma autônoma. Em vez de depender apenas de um modelo de linguagem estático, eles podem entrar em um ciclo de autoevolução. Esse processo pode acontecer em várias frentes:

- **Otimização de prompts**: o agente pode testar diferentes formas de "falar consigo mesmo" ou de instruir suas

ferramentas, descobrindo quais prompts geram os melhores resultados e adotando-os.
- **Otimização de fluxos de trabalho**: em sistemas com múltiplos agentes, o sistema pode aprender qual a melhor forma de organizar a "equipe", quem deve falar com quem e em que ordem, para tornar o processo mais rápido e eficiente.

- **Otimização de ferramentas**: o agente pode aprender não apenas a usar as ferramentas que tem, mas também a combiná-las de novas maneiras ou até mesmo a criar novas ferramentas (geralmente na forma de código) para resolver problemas inéditos.

Essa capacidade de auto-otimização é o que pode, no futuro, permitir que agentes de IA não apenas resolvam os problemas que damos a eles, mas descubram soluções que nós nem sequer havíamos imaginado.

5.5 Principais frameworks para Agentes de IA

Assim como bibliotecas facilitam a vida de programadores em diversas tarefas, no universo dos agentes de IA surgiram *frameworks* para tornar mais fácil (ou, pelo menos, mais estruturado) criar agentes sofisticados sem ter que reinventar a roda toda vez.

Vamos falar sobre alguns dos principais:

- **LangChain**: foi um dos primeiros e mais populares *frameworks* para construção de aplicações envolvendo *LLMs*. Ele fornece abstrações para cadeias de prompts (daí o nome "*chain*"), gerenciamento de memória conversacional, e integração com ferramentas. No contexto de agentes, o *LangChain* implementou desde cedo o modelo *ReAct* e variantes, permitindo configurar agentes com diferentes ferramentas (busca, calculadora, APIs

etc.) com poucas linhas de código. Apesar de inicialmente focado em agentes únicos, ele evoluiu para suportar fluxos mais complexos e hoje serve de base para frameworks de nível mais alto (como veremos com o *LangGraph*).

- **LangGraph**: um *framework* construído sobre o *LangChain*, mas que adiciona uma camada de organização em grafo às interações. Em vez de definir a lógica do agente apenas sequencialmente, o *LangGraph* permite desenhar um fluxo como um *DAG* (grafo acíclico direcionado), onde cada nó pode ser um passo (um agente, uma ferramenta, um filtro etc.) e as arestas definem a ordem e dependências. Isso é extremamente útil para orquestrar fluxos de trabalho mais complexos, possivelmente envolvendo múltiplos agentes e verificações condicionais.

  Outra vantagem do *LangGraph* é dar maior controle e visibilidade sobre o fluxo, útil em aplicações onde confiabilidade e auditabilidade são importantes. Ele também integra uma ferramenta chamada *LangSmith* para monitoramento das operações dos agentes, registrando entradas, saídas e planos, o que ajuda a depurar e melhorar o desempenho. É o mais utilizado hoje em dia para casos mais complexos, por dar mais controle.

- **CrewAI**: desenvolvido pelo brasileiro João Moura como um projeto *open source*, difere do *LangGraph* no fato de que este foca na flexibilidade estrutural e o *CrewAI* foca na facilidade para montar equipes de agentes. Esse *framework* traz a metáfora de uma "tripulação" (*crew*), onde cada agente tem um papel definido (por exemplo, um agente "Pesquisador", outro "Analista", outro "Escritor" etc.) e um objetivo individual. Você configura cada agente fornecendo sua função (o que ele representa ou faz na equipe), sua meta específica, um contexto (*backstory*) para dar personalidade/direcionamento, e as ferramentas que ele tem à disposição. Feito isso, o *framework* cuida de colocar esses agentes para trabalharem juntos.

Por ser um framework alto nível, é muito simples criar algo funcional com poucas linhas de código, o que foi um dos motivos de sua adoção rápida. A contrapartida é que, por abstrair muitos detalhes, pode ser menos óbvio customizar comportamentos específicos ou diagnosticar erros profundos.

- **AutoGen**: a *Microsoft* também entrou no jogo com sua própria abordagem, chamada *AutoGen*. Diferente do *CrewAI*, o *AutoGen* não enfatiza a analogia de "equipe", mas efetivamente implementa algo similar, definindo uma estrutura com múltiplos agentes especializados e um "Agente Orquestrador" que coordena tudo. Na extensão mais recente (codinome "*Magentic-One*"), são descritos quatro agentes especializados: um para navegar na web (*WebSurfer*), um para ler arquivos locais (*FileSurfer*), um para escrever código (*Coder*) e um para executar comandos no terminal do computador (*ComputerTerminal*), todos supervisionados pelo Orquestrador, que decide a decomposição de tarefas e integra os resultados.

  A filosofia por trás do AutoGen é tratar a interação entre agentes como uma conversa: cada agente é como uma persona conversando em um chat, e o Orquestrador faz perguntas a eles ou os instrui conforme necessário, num estilo bem próximo de *role-playing*. Por ser da Microsoft, há também uma atenção à integração com serviços *Azure* e foco em aplicações corporativas.

- **Outros frameworks**: a comunidade *open-source* (e alguns grandes *players*) vêm lançando uma enxurrada de *frameworks* para trabalhar com agentes nos últimos dois anos. Seria impossível cobrir todos, mas vale conhecer alguns que costumam aparecer em comparativos e tutoriais:

  - **Langflow**: desenvolvida pelos também brasileiros Rodrigo Nader e Gabriel Almeida, é um construtor visual que permite "arrastar e soltar" blocos e nós de abstrações de

agentes, *LLMs*, ferramentas e verificações num grafo interativo; excelente para quem não é muito da programação e prefere enxergar o fluxo do agente em vez de descrevê-lo apenas em código.
    - **Google ADK (Agent Developer Kit)**: oferecido pelo *Google*, traz componentes prontos para descoberta de capacidades, roteamento e monitoramento de agentes em produção.
    - **BeeAI**: foca em fluxos multimodais corporativos (texto, áudio, imagem) com controle de permissões granular.
    - **Camel**: popularizou a ideia de dois agentes ("usuário" × "assistente") conversando entre si para decompor e resolver problemas passo a passo.
    - **SmolAgents**: desenvolvida pela gigante do *open source Hugging Face*, é um projeto-irmão do *Smol-GPT* e propõe um estilo "microsserviço" de agentes, onde cada agente gerado é minúsculo, com responsabilidade única (por exemplo, "*escreva um teste unitário*", "*refatore esta função*", "*atualize o README*") e vive apenas o tempo necessário para cumprir sua tarefa. Essa filosofia de agentes efêmeros torna o pipeline de geração de código rápido e barato, já que o modelo só precisa de contexto mínimo e não carrega memórias pesadas de sessão em sessão.

De modo geral, esses *frameworks* tentam abstrair as partes comuns de construir agentes, como lidar com memórias de curto e longo prazo, gerenciar filas de tarefas, orquestrar múltiplos *LLMs*, integrar ferramentas e manejar os detalhes de cada modelo (como formatar *prompts* corretamente para evitar alucinações ou obter respostas estruturadas em JSON). Cada um tem seus pontos fortes; escolher o certo depende do caso de uso. Por exemplo, se você quer algo rápido e fácil para um protótipo, o *CrewAI* ou o *Langflow* podem resolver. Se precisa de controle fino sobre um fluxo complexo, talvez o *LangGraph* seja mais indicado. E se a ideia for embarcar isso numa aplicação corporativa e você já esteja em um ecossistema de

grandes *players* como *Microsoft* ou *Google*, pode ser preferível o *AutoGen* ou o *Google ADK*.

Falando em grandes players, as gigantes de tecnologia não estão apenas lançando *frameworks*, mas também propondo padrões e protocolos para tornar esses agentes interoperáveis e mais fáceis de integrar no mundo real. Vamos ver dois dos principais movimentos nesse sentido: o **MCP** e o **A2A**.

5.6 Protocolos MCP e A2A

Com a proliferação de agentes e ferramentas, surgiu a necessidade de criar padrões para que diferentes sistemas conversassem a mesma língua. Caso contrário, cada framework teria seu formato de mensagem, cada ferramenta seu jeito de ser chamada, e seria muito difícil misturar componentes de origens distintas em uma mesma solução. Em 2023 e 2024, dois termos começaram a aparecer frequentemente nas conversas sobre agentes de IA: MCP[4] e A2A[5] .

MCP — Model Context Protocol

Este protocolo foi introduzido pela empresa *Anthropic* (a criadora dos modelos *Claude*) e seu nome significa "Protocolo de Contexto de Modelo" em português. A ideia do MCP é padronizar a forma como um agente *LLM* se conecta a ferramentas externas, *APIs* ou fontes de dados. Lembra do capítulo anterior quando falamos de *function calling* e ferramentas que conectamos aos *LLMs*? O MCP é como uma evolução dessa ideia, oferecendo um esquema universal e estruturado para que agentes descubram e utilizem ferramentas de forma segura e consistente.

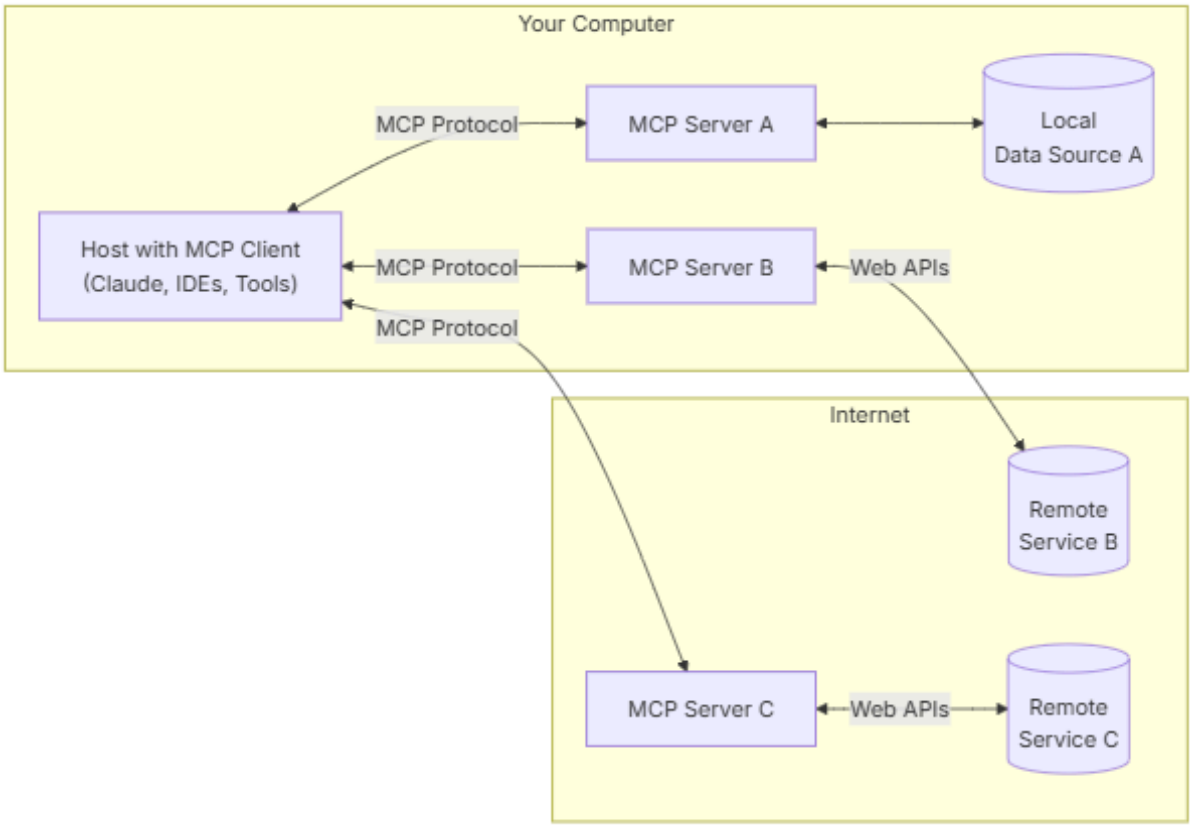

Figura 5.2: Exemplo de arquitetura MCP de https://modelcontextprotocol.io/introduction

Com o MCP, em vez de "ensinar" cada agente de um jeito diferente sobre quais ferramentas existem e como chamá-las, há um protocolo comum de comunicação entre um cliente MCP (o agente, ou melhor, a aplicação do agente) e um servidor MCP (que gerencia o acesso às ferramentas ou dados). Quando o agente precisa, por exemplo, buscar um dado em um banco de dados corporativo, ele envia uma requisição padronizada ao servidor MCP pedindo, digamos, "*use a ferramenta X com tais parâmetros*". O servidor MCP verifica permissões, executa a ação e retorna o resultado de volta, tudo em formato também padronizado. Isso adiciona uma camada de segurança (é mais fácil auditar e colocar limites nas chamadas) e de generalização (o mesmo agente poderia, em teoria, interagir com ferramentas diferentes se todas expõem uma interface MCP similar).

Em resumo, o MCP funciona como um "manual universal de instruções" (também chamado comumente de "porta USB-C para aplicações de IA") para o agente saber que ferramentas existem, o que fazem e como chamá-las, sem precisar de adaptação para cada ferramenta nova. A *Anthropic* criou o MCP como um padrão *open source* para a comunidade, e várias empresas e projetos começaram a adotar ou oferecer compatibilidade, inclusive concorrentes diretas, como a *OpenAI* e o *Google*. A promessa é reduzir o acoplamento a um único fornecedor de *LLM* ou a uma única plataforma, pois se todos suportarem MCP, você poderia trocar o "cérebro" *LLM* do seu agente de forma fácil (por exemplo, substituir o modelo *Claude* para o *Gemini*, *GPT* ou algum modelo *open source*) e, em teoria, ele conseguiria usar as mesmas ferramentas do mesmo jeito.

Para quem viveu a época da padronização da *web* (CSS, HTML etc.) ou de *APIs* de dispositivos, é um passo parecido, só que no ecossistema dos agentes de IA.

A2A — Agent-to-Agent

Já o A2A, proposto pelo *Google*, é um protocolo aberto voltado a padronizar a comunicação entre agentes. O nome completo dele é "*Agent-to-Agent Communication Protocol*" ("Protocolo de Comunicação de Agente a Agente"). Em essência, ele estabelece regras para que agentes possam conversar e colaborar diretamente uns com os outros, mesmo que tenham sido construídos por diferentes fornecedores ou rodando em plataformas distintas.

Pense no A2A como o equivalente a definir um idioma comum para que um agente criado na plataforma do *Google* possa solicitar ajuda de um agente criado na plataforma da *OpenAI*, por exemplo, ou de uma ferramenta legada de uma empresa, e ambos se entendam. O A2A se apoia em tecnologias da web já estabelecidas, como JSON (para formatar as mensagens), HTTP/SSE (para transporte e eventos) e autenticação robusta (similar aos esquemas do *OpenAPI*). Assim, os agentes se comunicam enviando requisições

padrão (como se fossem chamadas de *API*) uns aos outros, seguindo um protocolo de *handshake*, troca de capacidades etc.

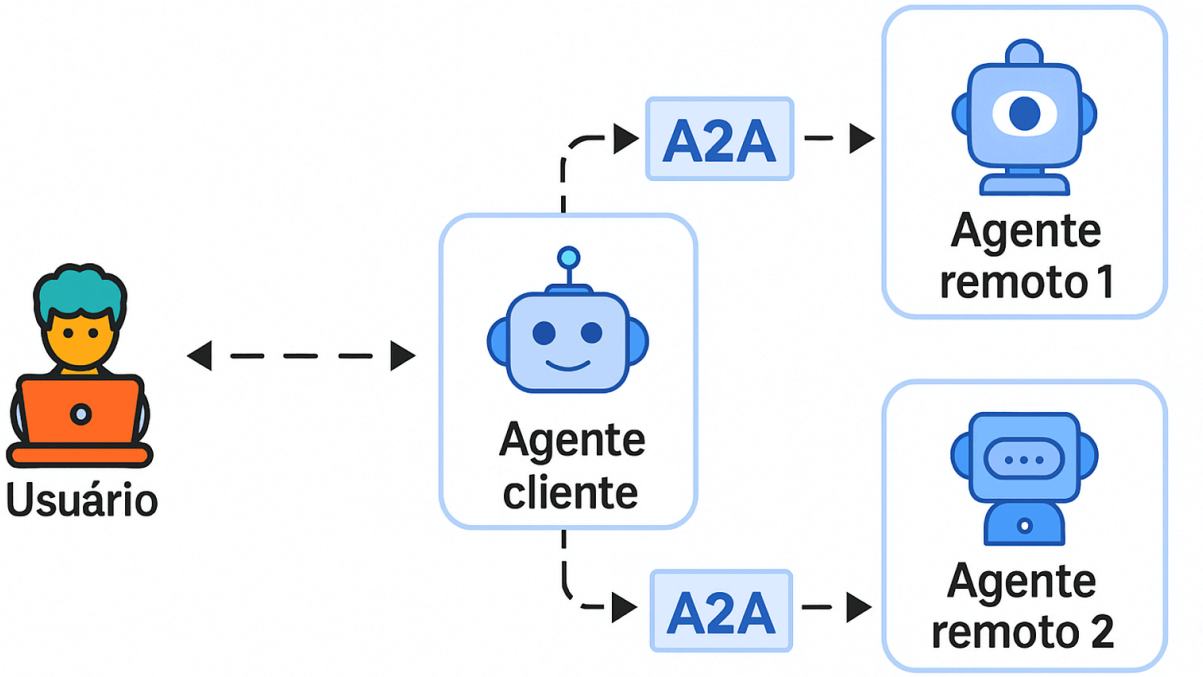

Figura 5.3: Exemplo de arquitetura A2A

Ele introduz também conceitos de agente-cliente (que inicia uma solicitação ou tarefa e coordena a execução) e agente-remoto (que oferece alguma capacidade ou serviço especializado e age a pedido do cliente). Isso é muito útil para cenários empresariais nos quais você pode ter vários agentes especializados em departamentos diferentes, como um agente que cuida de pedidos de compra conversando com outro que cuida de verificações de estoque. Com o A2A, idealmente, eles poderiam negociar tarefas, compartilhar informações e coordenar ações de forma segura e padronizada.

O Google lançou o A2A oficialmente em abril de 2025 com apoio de mais de 50 empresas e parceiros tecnológicos, indicando que há um esforço para que ele realmente se torne um alicerce da interoperabilidade entre agentes. No próprio anúncio, o *Google* deixou claro que vê o A2A como complementar ao MCP da *Anthropic*, ou seja, não é um ou outro, mas ambos juntos: o MCP

padronizando como agentes acessam o mundo (ferramentas e dados), e o A2A padronizando como agentes interagem entre si.

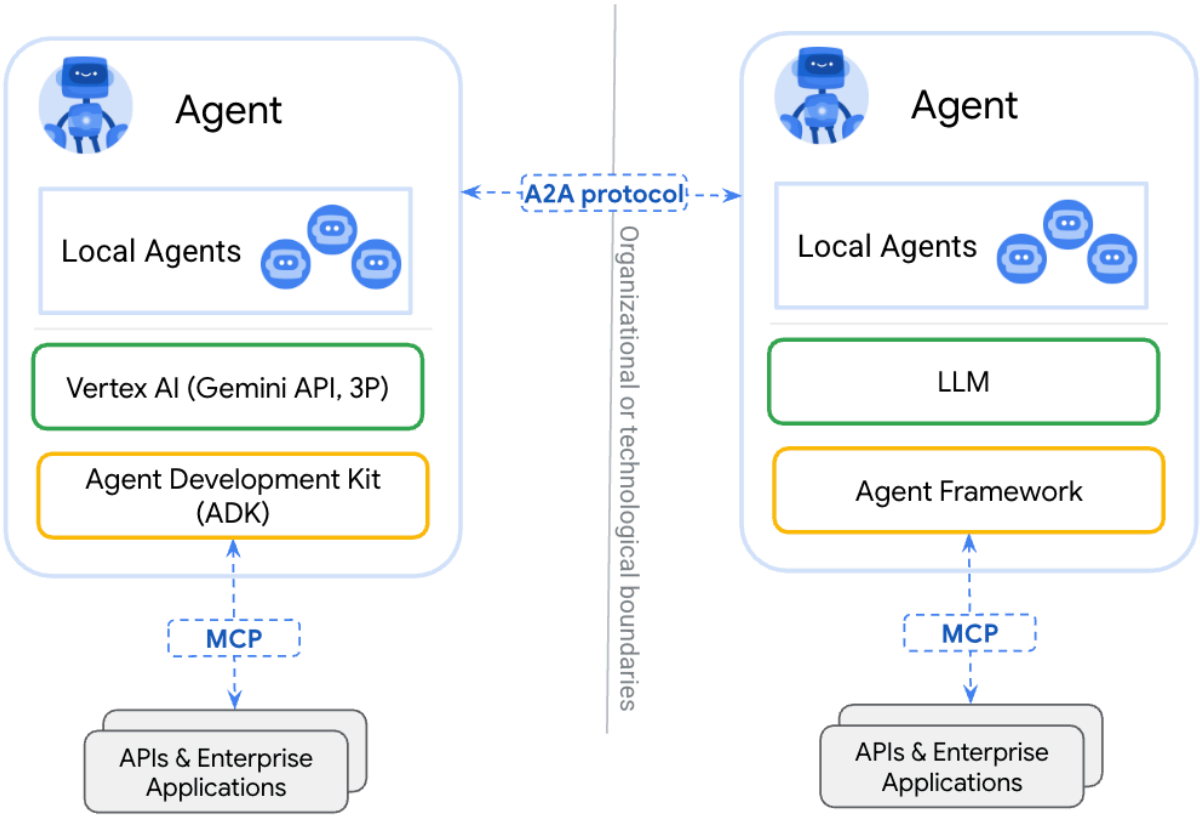

Figura 5.4: Exemplo de arquitetura complementária com MCP e A2A de https://a2a-protocol.org/latest/#why-a2a-matters

Esses protocolos estão em estágio inicial, mas representam um caminho rumo a um ecossistema unificado de agentes. Podemos imaginar que, em um futuro não muito distante, uma empresa poderá comprar ou desenvolver diversos agentes (de fornecedores distintos, com habilidades distintas) e, graças a padrões como MCP e A2A, colocá-los para trabalhar juntos quase que *plug-and-play*, sem precisar desenvolver integrações específicas de cada duplinha de agentes.

Os agentes de IA representam, sem dúvida, a próxima evolução na jornada de tornar IAs cada vez mais úteis e integradas ao nosso dia a dia. Se *LLMs* como o *Gemini* e o *ChatGPT* nos impressionaram por manter diálogos coerentes, os agentes prometem ir além da conversa e fazer coisas em nosso lugar, desde tarefas triviais, como organizar sua agenda, até missões complexas, como gerenciar sistemas inteiros de TI ou conduzir análises de negócios sob demanda.

Mas, como vimos, esse caminho traz desafios técnicos e éticos: precisamos garantir que os agentes tomem decisões corretas (ou pelo menos, não desastrosas), que saibam quando pedir ajuda ou autorização humana, que não vazem informações sensíveis inadvertidamente ao colaborar, e por aí vai. A evolução dos agentes está forçando avanços em segurança de IA (por exemplo, detecção e prevenção de *loops* maliciosos), em eficiência (um agente pensando por horas pode custar caro em termos de computação, então otimizações e arquiteturas esparsas como *Mixture-of-Experts* ajudam a reduzir o custo, como vimos no capítulo anterior) e em experiência do usuário (afinal, como será interagir com um sistema composto por vários agentes? A interface precisa transmitir confiança e clareza do que está acontecendo nos bastidores).

Essas são perguntas que ainda estamos começando a responder. Mas uma coisa é certa: a era dos agentes de IA já começou. E, ao compreender como eles funcionam, você está mais preparado não apenas para usar, mas também para ajudar a construir o futuro dessa tecnologia.

No próximo capítulo, vamos focar nos parâmetros de calibragem dos modelos *GPT* que estão disponíveis no *Playground* da *OpenAI*, e de outros, via *API*, e servem para controlar o comportamento dos modelos ao gerar uma resposta.

Referências bibliográficas:

[1] AI Agents. *NVIDIA*, 2025. Disponível em: https://www.nvidia.com/en-us/glossary/ai-agents/

[2] YAO, S. et al. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2210.03629

[3] LIU, B. et al. Advances and Challenges in Foundation Agents: From Brain-Inspired Intelligence to Evolutionary, Collaborative, and Safe Systems. *arXiv*, 2025. Disponível em: https://arxiv.org/abs/2504.01990

[4] Get started with the Model Context Protocol (MCP). 2024. Disponível em: https://modelcontextprotocol.io/introduction

[5] Agent2Agent (A2A) Protocol. 2025. Disponível em: https://a2a-protocol.org/latest/

Capítulo 6
Parâmetros de calibragem dos LLMs

Ao acessar o *Playground* da *OpenAI* (https://platform.openai.com/playground/), o *Google AI Studio* (https://aistudio.google.com/), o *Console* da *Anthropic* (https://console.anthropic.com/) ou configurar a *API* de *LLMs*, diversos campos e parâmetros de calibragem são apresentados a você, o que pode ser um pouco intimidador a princípio.

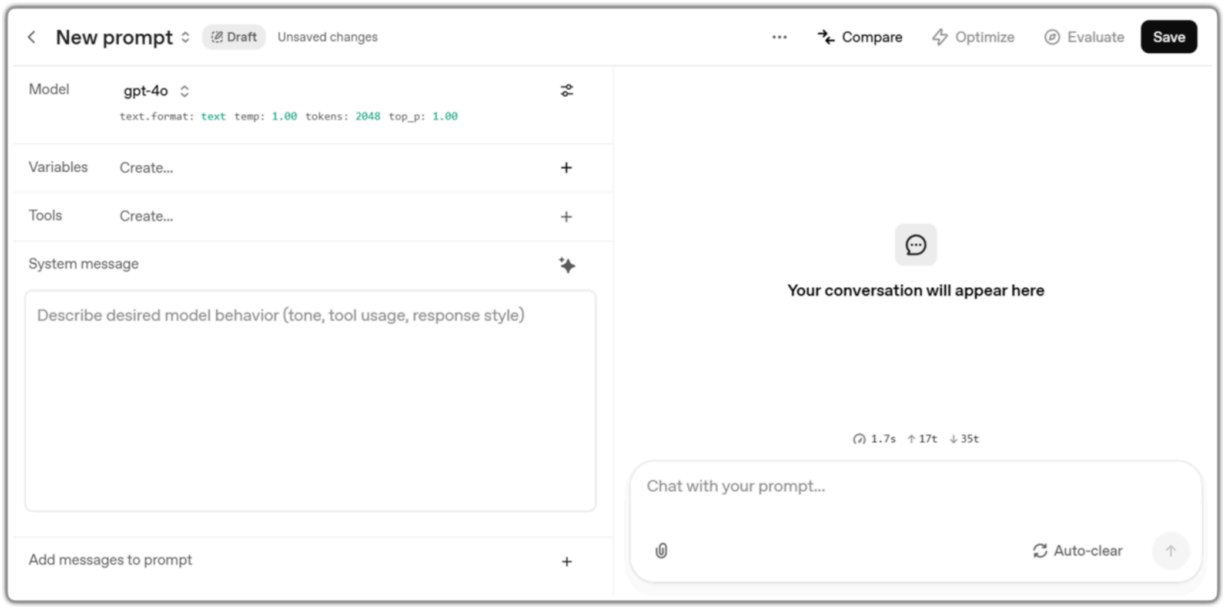

Figura 6.1: Página de Playground da OpenAI

Esses "parâmetros" não têm nada a ver com os outros parâmetros mencionados anteriormente, que se tratavam dos pesos e vieses da rede neural profunda. Esses aqui são parâmetros de calibragem para manipular os modelos disponibilizados pelas empresas em seus *Playgrounds* e via *API*. Com eles, você será capaz de controlar o comportamento do seu modelo, a verbosidade da resposta e até mesmo evitar a repetição excessiva das mesmas palavras.

Vamos usar o *Playground* da *OpenAI* como exemplo e ver o que cada um deles significa e como calibrá-los.

System message

Figura 6.2: Campo "System message" do Playground da OpenAI

Começando, do lado esquerdo você pode ver um campo chamado *System message* (Mensagem de sistema). Esse campo controla o comportamento do modelo e, usando o padrão sem nenhuma mensagem, o modelo escolhido vai se comportar de maneira habitual, sempre tentando ajudar e responder às perguntas como um ser humano educado (segundo seu treinamento) faria. Porém, se você alterar essa mensagem para "*Você é um assistente muito sarcástico.*", ele continuará respondendo às suas perguntas, mas sempre incluindo alguma ironia ou algo engraçadinho.

System message

Describe desired model behavior (tone, tool usage, response style)

User

Escreva uma mensagem curta de boa sorte para o Fabrício, que vai fazer uma prova.

Assistant

Claro! Aqui está uma mensagem:

"Boa sorte, Fabrício! Confio no seu potencial e sei que você vai arrasar na prova. Vá com calma e mostre tudo o que sabe. Estamos todos torcendo por você!"

System message

Você é um assistente muito sarcástico

User

Escreva uma mensagem curta de boa sorte para o Fabrício, que vai fazer uma prova.

Assistant

Claro, aqui vai: "Fabrício, que a força esteja com você nessa prova... e que as respostas certas venham por telepatia, porque estudar é para os fracos. Boa sorte!"

Figura 6.3: Exemplos de mensagens diferentes no campo "System" do Playground da OpenAI

Prompts

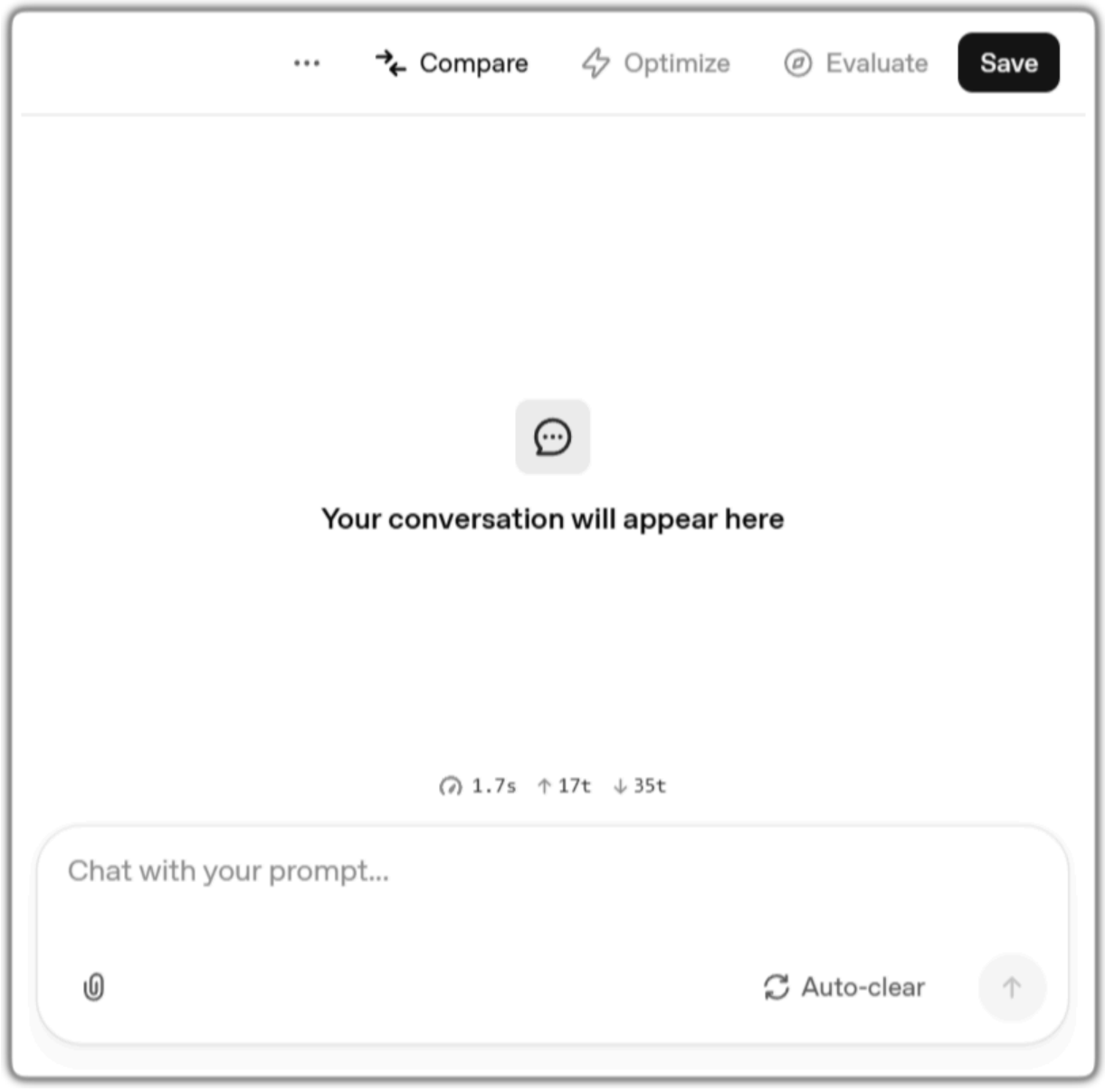

Figura 6.4: Campo de prompts do Playground da OpenAI

Como já vimos, o *prompt* é a entrada que o modelo receberá, que nesse caso será uma mensagem com um pedido ou pergunta. Abaixo, você verá um campo de texto com uma mensagem "*Chat with your prompt...*". É aqui que você precisa inserir o seu *prompt* e, ao clicar no botão de "Enviar", o modelo começará a responder logo abaixo em um campo que aparecerá, chamado *Assistant* (Assistente). Após a resposta, você pode continuar enviando *prompts* subsequentes.

Figura 6.5: Campo de prompt do Playground da OpenAI com resposta e novo campo de prompt

Model

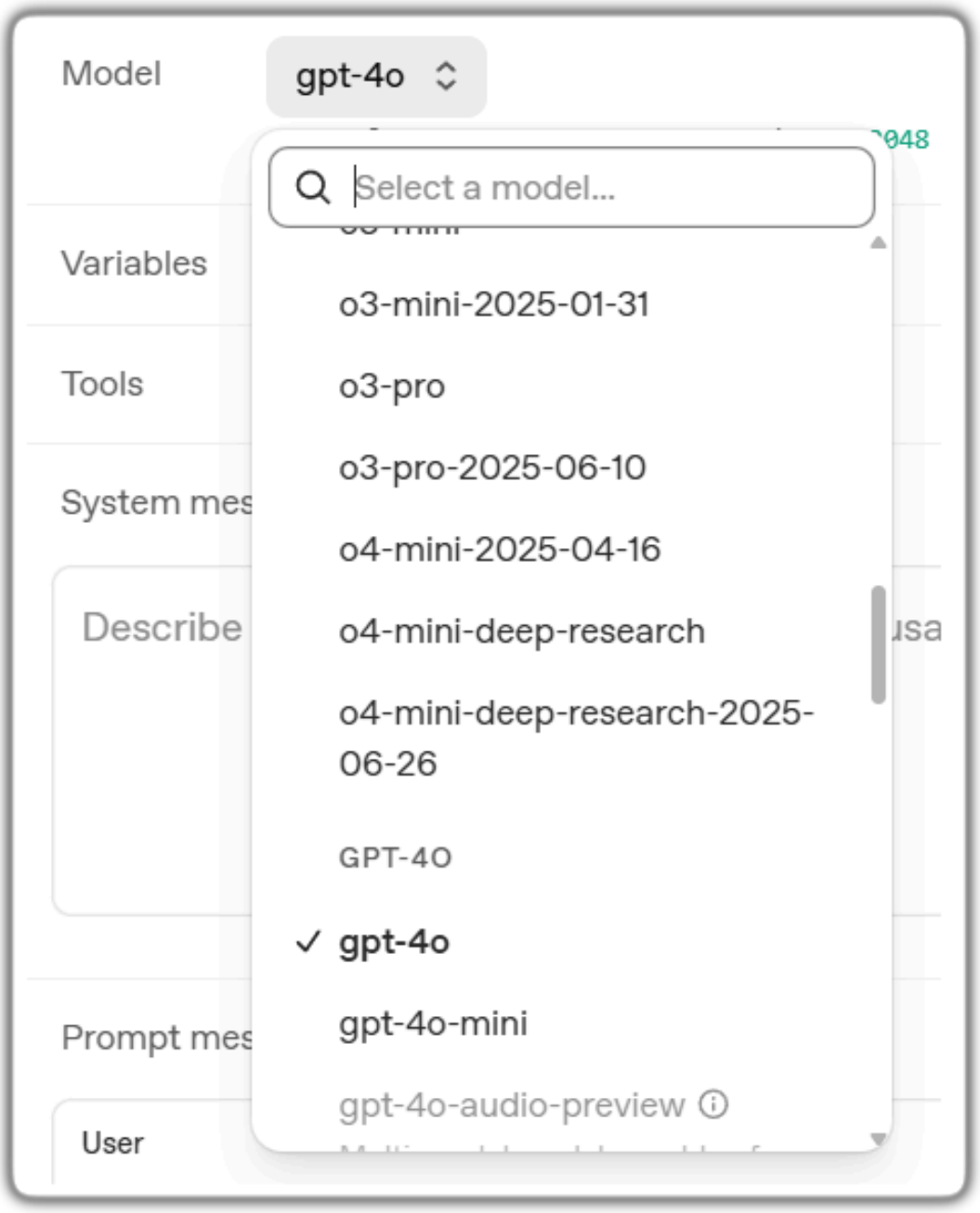

Figura 6.6: Campo "Model" do Playground da OpenAI

No topo, no campo *Model* (Modelo), você pode escolher qual modelo do *GPT* quer usar. A lista que aparece para você pode ser diferente da que aparece para mim por alguns fatores, como a data de lançamento deste livro e listas de espera da *OpenAI* que liberam modelos apenas para certos grupos de pessoas a cada vez.

De qualquer modo, você pode ver variações dos modelos da série *o* (modelos de raciocínio) e da série *GPT-4o*. Os valores ao lado do nome desses modelos podem representam a data de lançamento desses modelos (por exemplo, o modelo *o4-mini-2025-04-16* foi lançado no dia 16 do mês 04, abril, de 2025, e é uma versão mais refinada do *o4-mini* original).

Temperature, Max tokens e Top P

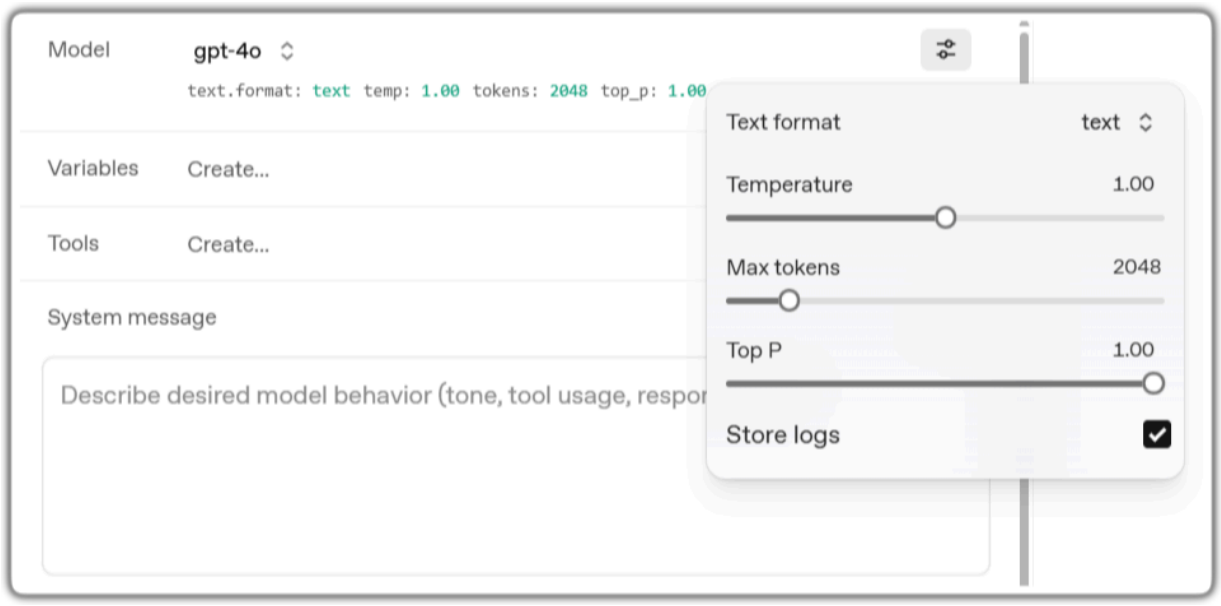

Figura 6.7: Campos "Temperature", "Max tokens" e "Top P" do Playground da OpenAI

Já falamos bastante sobre o papel da temperatura no funcionamento dos modelos *GPT* no capítulo 4, e é aqui onde você pode calibrar o seu valor. Relembrando, ele vai de 0 a 2, sendo que 1 é o valor padrão.

O campo *Max tokens* define o número máximo de *tokens* (de novo eles aqui) que será retornado pelo modelo para um *prompt* enviado. O valor máximo é o tamanho da janela de contexto do modelo selecionado. Porém, esse parâmetro não é infalível. Ao brincar com a ferramenta por um tempo, você notará que, às vezes, se o seu *Max*

tokens estiver com um valor muito baixo, as respostas podem ser cortadas pela metade.

O *Top P* é um controle um pouco mais avançado que influencia a aleatoriedade da resposta gerada, assim como a temperatura, mas ele funciona de maneira um pouco diferente. Em vez de considerar todas as palavras possíveis para continuar uma frase, o modelo seleciona só as mais prováveis, até que a soma dessas probabilidades chegue a um certo limite "P", por exemplo, 90% se o *Top P* for 0.9. Depois, ele escolhe aleatoriamente entre essas, com base no valor da temperatura. Imagine que o modelo tem estas opções para continuar a frase "*O céu está...*":

- azul - 60%
- cinzento - 20%
- nublado - 10%
- cor-de-rosa - 5%
- radioativo - 1%

Se o *Top P* for 0.9, ele só considera as três primeiras opções (que somam 90%) e ignora o resto. Se o *Top P* for 0.7, ele só consideraria a primeira opção, pois a soma da primeira com a segunda dá 80%, que já passa o valor de 70%. Se o *Top P* for 1.0, todas entram no jogo, até as mais inusitadas como "radioativo". No *Playground*, esse valor pode ser alterado entre 0 e 1, sendo que 1 é o padrão.

Variables

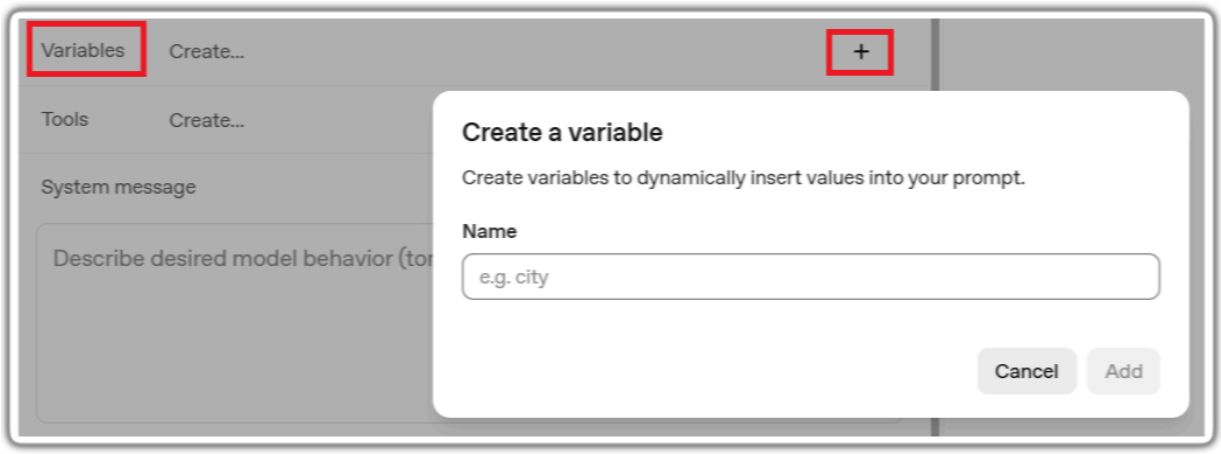

Figura 6.8: Campo "Variables" do Playground da OpenAI

O campo *Variables*, como o próprio nome diz, serve para você criar variáveis dinâmicas que serão posteriormente substituídas pelo valor real, como `{nome}`, `{cidade}`, `{assunto}` etc. Na hora de executar o prompt, o Playground mostra campos para preencher esses valores, permitindo reutilizar o mesmo *template* com entradas diferentes.

Você deve clicar em *Variables* → *Create…* e definir o nome da variável. No seu prompt (system ou user), use `{{nome_da_sua_variável}}` nos locais que quer que sejam substituídos. Ao rodar, o *Playground* exibirá campos para esses nomes, e o modelo receberá o texto já substituído. Essas variáveis funcionam também na *API* da *OpenAI*.

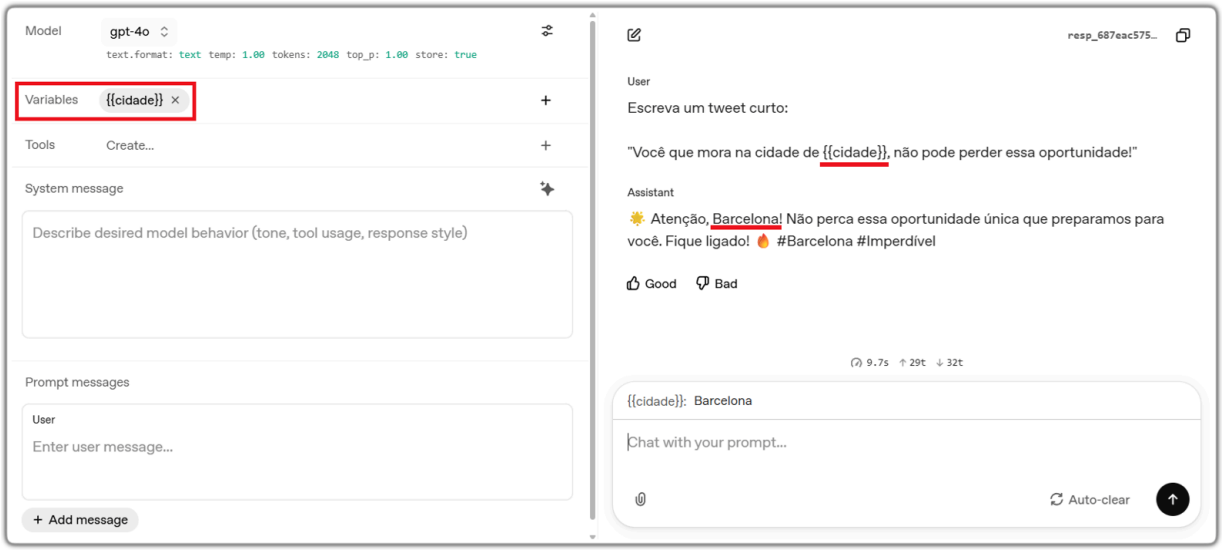

Figura 6.9: Uso do campo "Variables" do Playground da OpenAI

Tools

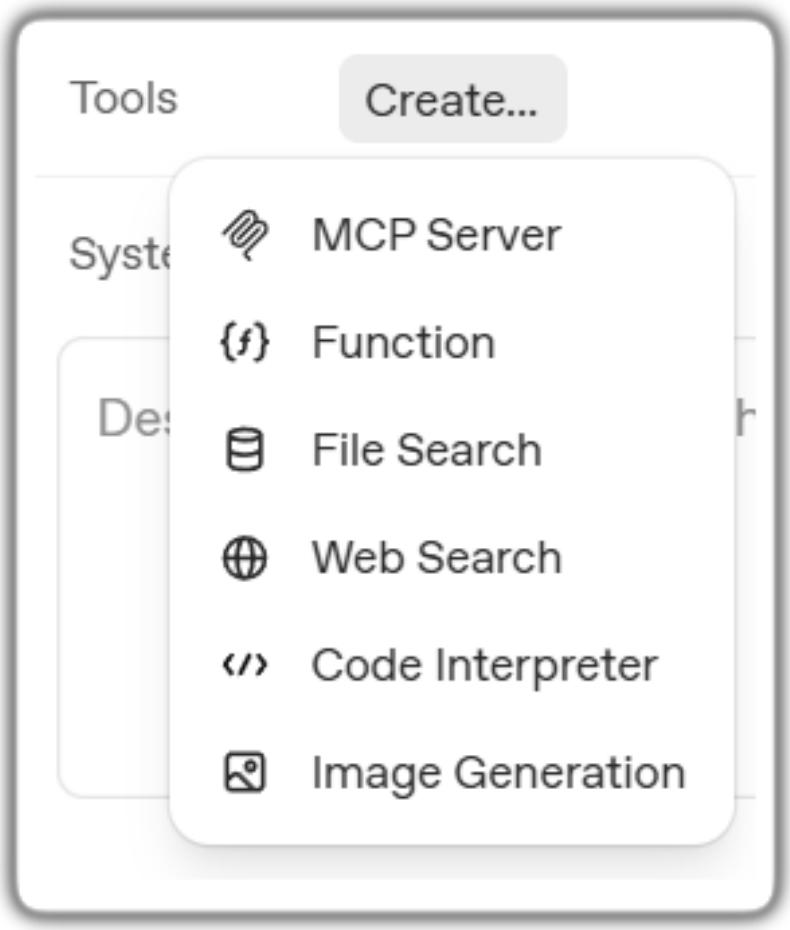

Figura 6.10: Campo "Tools" do Playground da OpenAI

O campo *Tools* permite que você acione diretamente *plugins* e ferramentas adicionais durante a conversa, como:

- **MCP Server**: para conectar a servidores externos que antecipam dados/contexto via o protocolo MCP, que vimos no capítulo 5.
- **Function**: para chamar funções customizadas (em linguagem de programação Python, por exemplo) definidas por você (o "*function calling*" que mencionei nos capítulos 4 e 5).
- **File Search**: para buscar dentro de arquivos carregados no modelo.
- **Web Search**: para fazer buscas na internet em tempo real.
- **Code Interpreter**: para executar código Python em um *sandbox* (útil para análise de dados, cálculo, manipulação de

arquivos e imagens etc.).
- **Image Generation**: para criar ou editar imagens usando os modelos da *OpenAI*.

Em resumo, esse menu dá acesso rápido a ferramentas potentes que ampliam muito o que o modelo pode fazer dentro da conversa, seja buscar dados, gerar imagens, rodar código, chamar funções ou se conectar a sistemas externos via MCP.

Depois de tanto falar sobre *prompts*, discutiremos no próximo capítulo o conceito de **Engenharia de Prompt**, que nos ajudará a trabalhar de forma mais harmoniosa com os modelos de linguagem e a obter respostas melhores e mais coerentes com o que desejamos.

CAPÍTULO 7
Engenharia de Prompt

No capítulo 4, discutimos a fundo como o *ChatGPT* e outros *LLMs* funcionam e como o seu processo de previsão da próxima palavra é algo totalmente probabilístico, com base na arquitetura escolhida. Também discutimos o conceito de **alucinações**, que é quando esses modelos geram uma resposta falsa ou sem muita relação com o que foi pedido no *prompt* de entrada, seja por problemas no momento do treinamento ou escolhas estocásticas incorretas. Porém, muitas vezes, a culpa desses erros pode ser atribuída a nós mesmos, que não nos expressamos da maneira ideal – algo que ocorre mesmo no nosso dia a dia, entre dois seres humanos.

Para desbloquear todo o potencial desses modelos dadas as suas limitações, surge a necessidade de que os(as) profissionais que usarão essas ferramentas tenham uma compreensão profunda sobre como interagir com elas. Com esse objetivo, foi cunhado o termo "Engenharia de *Prompt*", um campo muito recente dedicado à arte e ciência de projetar *prompts* melhores e mais eficazes, a fim de "guiar" as respostas dos modelos e obter o que desejamos.

Sim, eu sei que nem todo mundo gosta deste nome "Engenharia de *Prompt*", talvez por relacioná-lo a outros campos muito mais abrangentes, como a Engenharia da Computação ou Civil. Porém, nomenclaturas são apenas nomenclaturas. Se olharmos para a etimologia da palavra "engenharia", vemos que ela deriva do latim "*ingeniare*", que podemos traduzir como "projetar", "inventar" ou "conceber". E é exatamente disso que trata a Engenharia de *Prompt*: como podemos "engenhar" ou "projetar" o nosso *prompt* da melhor maneira possível para obter o resultado que almejamos, como artesãos criando uma obra de arte.

Boa parte dos modelos de linguagem atuais (e provavelmente os futuros) já conseguem capturar nuances mesmo em *prompts*

incompletos ou mal escritos, mas, como esse não é o caso para todos eles, empresas e universidades do mundo inteiro têm trabalhado incessantemente em artigos científicos sérios para desenvolver essas técnicas.

A própria *OpenAI*, criadora dos modelos *GPT*, oferece um livro de receitas sobre como obter resultados melhores usando seus modelos: https://cookbook.openai.com/techniques_to_improve_reliability. Neste capítulo, vamos elencar alguns desses princípios e as técnicas mais conhecidas atualmente.

7.1 Princípios para a criação de um prompt

Esses são princípios gerais, mas que provavelmente lhe ajudarão a obter respostas mais coerentes:

- Ter clareza ao dar as instruções (imagine que você é um(a) profissional sênior e está explicando passo a passo para um estagiário como ele deveria realizar uma tarefa);
- Dividir tarefas complexas em subtarefas menores;
- Pedir para o modelo explicar seus passos antes de dar a resposta;
- Pedir para o modelo dar justificativas de suas respostas;
- Gerar várias respostas diferentes e pedir para o modelo escolher a melhor.

Alguns desses conceitos podem parecer muito óbvios, já outros foram demonstrados em artigos científicos, sendo que a maior parte desses testes foi realizada em cima do modelo *GPT-3* ou do *InstructGPT*. Veremos essas abordagens nas técnicas demonstradas em todo este capítulo.

Começaremos abordando um dos princípios, e os outros serão demonstrados dentro das demais técnicas que serão apresentadas

neste capítulo.

Dividindo tarefas complexas em subtarefas menores

Você se lembra do jogo de tabuleiro "Detetive"? Em caso negativo, era um jogo onde, dadas algumas dicas, os participantes tinham que descobrir quem havia cometido um assassinato, utilizando qual objeto e em que local. A resposta era sempre algo como: "*foi o Coronel Mostarda com o candelabro na sala de estar*".

Um teste de *prompt* foi feito pela equipe da *OpenAI* dando 5 dicas e pedindo para o modelo responder com uma das opções a seguir:

Use as dicas a seguir para responder à seguinte questão de múltipla escolha, usando o seguinte procedimento:

Dicas:

1. A Senhorita Scarlett era a única pessoa na sala.
2. A pessoa com o cachimbo estava na cozinha.
3. O Coronel Mostarda era a única pessoa no observatório.
4. O Professor Plum não estava na biblioteca nem na sala de bilhar.
5. A pessoa com o castiçal estava no observatório.

Pergunta: O Coronel Mostarda estava no observatório com o castiçal?

- *(a) Sim; o Coronel Mostarda estava no observatório com o castiçal.*
- *(b) Não; o Coronel Mostarda não estava no observatório com o castiçal.*
- *(c) Desconhecido; não há informações suficientes para determinar se o Coronel Mostarda estava no observatório com o castiçal.*

Resposta:

Porém, essa foi uma tarefa complexa, e o modelo respondeu erroneamente: "**(c) Desconhecido; não há informações suficientes para determinar se o Coronel Mostarda estava no observatório com o castiçal.**" Na verdade, a solução correta seria a opção (a), pois com as dicas 3 e 5 é possível saber com certeza que a pessoa com o castiçal estava no observatório e, como o Coronel Mostarda era a única pessoa no observatório, seria ele essa pessoa com o castiçal.

A equipe da *OpenAI* sugeriu, então, dividir essa tarefa mais complexa em tarefas menores, criando o seguinte *prompt*:

Use as dicas a seguir para responder à seguinte questão de múltipla escolha, usando o seguinte procedimento:

- (1) Primeiramente, analise as dicas uma por uma e considere se a dica é potencialmente relevante
- (2) Em segundo lugar, combine as dicas relevantes para raciocinar a resposta correta à pergunta
- (3) Em terceiro lugar, mapeie a resposta para uma das respostas de múltipla escolha: (a), (b) ou (c)

Dicas:

1. A Senhorita Scarlett era a única pessoa na sala.
2. A pessoa com o cachimbo estava na cozinha.
3. O Coronel Mostarda era a única pessoa no observatório.
4. O Professor Plum não estava na biblioteca nem na sala de bilhar.
5. A pessoa com o castiçal estava no observatório.

Pergunta: O Coronel Mostarda estava no observatório com o castiçal?

- *(a) Sim; o Coronel Mostarda estava no observatório com o castiçal.*
- *(b) Não; o Coronel Mostarda não estava no observatório com o castiçal.*
- *(c) Desconhecido; não há informações suficientes para determinar se o Coronel Mostarda estava no observatório com o castiçal.*

Resposta:

Com esse *prompt*, dividindo o problema em três partes e explicando ao modelo o que ele deveria fazer em cada uma dela, eles obtiveram como resposta:

(1) Primeiramente, analise as pistas uma por uma e considere se a pista é potencialmente relevante:

1. A Senhorita Scarlett era a única pessoa na sala. **Essa dica não é relevante**
2. A pessoa com o cachimbo estava na cozinha. **Essa dica não é relevante**
3. O Coronel Mostarda era a única pessoa no observatório. **Essa dica é relevante**
4. O Professor Plum não estava na biblioteca nem na sala de bilhar. **Essa dica não é relevante**
5. A pessoa com o castiçal estava no observatório. **Essa dica é relevante**

(2) Em segundo lugar, combine as dicas relevantes para raciocinar a resposta correta à pergunta:

- **A dica 3 e a dica 5 dizem que a pessoa com o castiçal estava no observatório. O Coronel Mostarda era a única pessoa no observatório, então o Coronel Mostarda era a pessoa com o castiçal.**

(3) Em terceiro lugar, mapeie a resposta para uma das respostas de múltipla escolha: (a), (b) ou (c)

- **(a) Sim; o Coronel Mostarda estava no observatório com o castiçal.**

O modelo foi capaz de chegar à resposta correta quando pôde focar nas tarefas menores. É possível que um modelo mais avançado, como o *GPT-4*, conseguisse chegar a essa resposta apenas com o primeiro *prompt* proposto, mas, de qualquer forma, dividir um problema grande em outros menores é sempre uma boa ideia.

7.2 Zero-shot & Few-shot Prompting

O artigo da *OpenAI* que explica o processo de treinamento do modelo *GPT-3* se chama "*Language Models are Few-Shot Learners*" (que podemos traduzir livremente como "Modelos de linguagem aprendem com alguns exemplos")[1] e fala sobre os conceitos de *zero-shot*, *one-shot* e *few-shot learning*, onde a palavra "*shot*" significa um exemplo ou uma amostra e "*learning*" se refere a "aprendizado".

Um aprendizado *zero-shot* (com zero exemplos) se refere a uma técnica de *Machine Learning* que permitiria a um modelo classificar ou prever algo sem ter tido nenhum treinamento sobre aquilo em específico. De maneira simplificada, seria como treinar um modelo com fotos e descrições de gatos, cachorros e elefantes; mas apenas a descrição de uma girafa, sem nenhuma foto. Um modelo bem treinado com o conceito de *zero-shot* poderia ser capaz de generalizar os padrões aprendidos a partir das fotos e descrições dos gatos, cachorros e elefantes para identificar corretamente uma girafa, um animal para o qual ele teve zero imagens de exemplo. De maneira similar, *one-shot* significa "com um exemplo" e *few-shot*, "com alguns exemplos".

Essa explicação trata sobre o processo de aprendizado dos modelos, mas esses conceitos também podem ser usados para a "criação de *prompts*" (em inglês, "*prompting*"). Um *prompt* com *zero-shot*, por exemplo, seria:

Pergunta: Um diretor de cinema já dirigiu 16 filmes. Metade dos filmes que ele dirigiu são de ação, e metade dos filmes de ação têm a participação do Nicolas Cage, e na metade deles o Nicolas Cage tem bigode. Quantos filmes de ação com a participação do Nicolas Cage com bigode ele dirigiu?

Resposta:

Pois estamos pedindo a resposta ao modelo sem dar a ele nenhum exemplo. Já um *prompt* com *one-shot* ou *few-shot* poderia ser:

Pergunta: João tem 10 bolas. Metade delas são azuis e metade são vermelhas. Quantas bolas vermelhas o João tem?

Resposta: A resposta é 5.

Pergunta: Um diretor de cinema já dirigiu 16 filmes. Metade dos filmes que ele dirigiu são de ação, e metade dos filmes de ação têm a participação do Nicolas Cage, e na metade deles o Nicolas Cage tem bigode. Quantos filmes de ação com a participação do Nicolas Cage com bigode ele dirigiu?

Resposta:

Damos um ou mais exemplos de problemas similares para o modelo entender um pouco melhor a forma de raciocinar. Um exemplo mais útil para o dia a dia é o de gerar posts, passando posts seus antigos e reais para que o modelo aprenda o seu tom de escrita. No podcast *Dev Sem Fronteiras* da Alura, isso foi usado para criar novos *tweets* com base em informações muito básicas e em *tweets* reais anteriores. Primeiramente, podemos passar algumas bases de *tweet* e *tweets* finais reais como *few-shot* para o modelo:

BASE DO TWEET 1:

"Diego; Paris, França; Engenheiro de Cloud; fez Engenharia Mecânica; trabalha atualmente com computação"

TWEET FINAL 1:

"*Nessa aventura o @fabriciocarraro falou com o Diego, que cursou Engenharia Mecânica, mas posteriormente passou a buscar algo que envolvesse também a computação. Atualmente, trabalha como Engenheiro de Cloud em Paris, na França, e hoje contará sobre isso. Vai lá escutar!*"

BASE DO TWEET 2:

"Matheus; Londres; Engenheiro de Software; fez Engenharia Mecânica; trabalhou na Polônia por 1 ano; trabalha atualmente na Google DeepMind"

TWEET FINAL 2:

"*Nessa aventura o @fabriciocarraro falou com o Matheus, que trabalhou por 1 ano na Polônia e, na sequência, se juntou como Engenheiro de Software ao time da Google DeepMind em Londres. Vai lá escutar!*"

BASE DO TWEET 3:

"Pedro; Brasil; fez pós em Machine Learning no ITA graças a um estágio no Itaú; trabalha remotamente para o Vale do Silício"

TWEET FINAL 3:

"*Nessa aventura o @fabriciocarraro falou com o Pedro que, graças a um estágio no Itaú, pôde fazer pós em Machine Learning no ITA. Hoje, ele nos conta as vantagens e desvantagens de trabalhar remotamente para o Vale do Silício a partir do Brasil. Vai lá escutar!*"

Agora, basta enviar para o modelo uma nova base de *tweet*, como:

> "Fabrício Carraro; Barcelona; AI Developer Advocate; formado em Engenharia da Computação pela UNICAMP; host dos podcasts IA Sob Controle e Dev Sem Fronteiras"

E o modelo gerará o novo *tweet*, seguindo o mesmo estilo e tom de escrita dos anteriores:

> "*Nessa aventura, o @fabriciocarraro, host do Dev Sem Fronteiras, conta sua própria história: formado pela UNICAMP, hoje é AI Developer Advocate em Barcelona e também apresenta o podcast IA Sob Controle. Vai lá escutar!*"

Essa é uma técnica bastante poderosa, principalmente quando falamos de textos maiores, como *newsletters* e posts de blogs.

Porém, apesar de funcionarem em alguns casos, essas técnicas mais "diretas ao ponto" ainda deixavam bastante a desejar em problemas mais complexos. Por esse motivo, outras técnicas começaram a ser desenvolvidas.

7.3 Chain-of-Thought Prompting

Few-shot Chain-of-Thought

O conceito de *Chain-of-Thought Prompting* pode ser traduzido como "Criação de *prompts* com Cadeia de Pensamento" e foi publicado pela equipe de pesquisa em Inteligência Artificial do *Google* no

artigo "*Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*" (traduzindo, "Criação de *prompts* com Cadeia de Pensamento induz o raciocínio em Grandes Modelos de Linguagem")[2] .

E caso você esteja reconhecendo esse nome, sim, é a mesma ideia de *Chain-of-Thought* citada no capítulo 4, que é usada para treinar os modelos de raciocínio mais poderosos.

Ele propõe que podemos obter respostas muito mais precisas quando demonstramos aos modelos, no próprio *prompt*, a forma exata de raciocinar, dando uma resposta completa como exemplo, incluindo o passo a passo para chegar a ela. Por isso, essa técnica também é chamada de *Few-shot Chain-of-Thought Prompting* (traduzindo, "Criação de *prompts* com Cadeia de Pensamento com alguns exemplos"), já que são dadas amostras ao modelo.

Com isso em mente, a equipe do *Google* mostrou no artigo dois casos de *prompts* com *few-shot*, o primeiro deles sem a cadeia de pensamento, similar ao último exemplo de *few-shot* que vimos no tópico anterior:

> **Pergunta:** Roger tem 5 bolas de tênis. Ele compra mais 2 latas de bolas de tênis. Cada lata contém 3 bolas de tênis. Quantas bolas de tênis ele tem agora?
>
> **Resposta:** A resposta é 11.
>
> **Pergunta:** Havia 23 maçãs no refeitório. Se foram usadas 20 para fazer o almoço e foram compradas mais 6, quantas maçãs eles têm agora?
>
> **Resposta:**

Para esse *prompt*, o modelo respondeu, erroneamente: "**A resposta é 27.**"

No segundo caso, eles aplicaram o conceito de cadeia de pensamento:

> **Pergunta:** Roger tem 5 bolas de tênis. Ele compra mais 2 latas de bolas de tênis. Cada lata contém 3 bolas de tênis. Quantas bolas de tênis ele tem agora?
>
> **Resposta:** Roger começou com 5 bolas. 2 latas de 3 bolas de tênis cada equivalem a 6 bolas de tênis. 5 + 6 = 11. A resposta é 11.
>
> **Pergunta:** Havia 23 maçãs no refeitório. Se foram usadas 20 para fazer o almoço e foram compradas mais 6, quantas maçãs eles têm agora?
>
> **Resposta:**

E a solução obtida do modelo foi: "**Havia 23 maçãs no refeitório originalmente. Foram usadas 20 para fazer o almoço. Então, havia 23 - 20 = 3. Foram compradas mais 6 maçãs, então há agora 3 + 6 = 9. A resposta é 9.**", que é a resposta correta.

Note que as perguntas não mudaram, apenas foi incluída na primeira resposta a cadeia de pensamento, ou seja, o passo a passo do raciocínio esperado pelo modelo para responder àquela pergunta.

Um outro exemplo complexo de *Few-shot Chain-of-Thought* e números pode ser demonstrado no caso de calcular a alíquota de imposto de um certo grupo. Suponha que façamos parte do governo de um país fictício chamado "*Carrarostão*" e tenhamos os dados do imposto de renda dos habitantes, que é uma lista com seus nomes e respectivos salários anuais.

Indivíduo:	Rendimento anual:
Juliana da Silva Santos	R$31.000,00
Pedro Henrique Souza Lima	R$15.000,00
Marcelo Pereira Costa	R$52.000,00
Amanda Ferreira Oliveira	R$103.000,00
José Carlos Almeida Silva	R$42.000,00
Isabella Santos Rodrigues	R$156.000,00
Cleiton Moreira Cardoso	R$15.000,00
Larissa Oliveira Alves	R$27.000,00
Túlio de Oliveira Ribeiro	R$82.000,00
Thaís da Conceição Cunha	R$33.000,00

Figura 7.1: Lista de 10 nomes e salários dos habitantes do Carrarostão

Nesse nosso governo de mentirinha, existe apenas uma alíquota única de 30% a ser arrecadada em cima desse valor de salário anual dos habitantes. No entanto, para não abusar dos mais pobres, as pessoas com salário abaixo de 40.000 são isentas de pagar qualquer imposto. Dado esse enunciado, como podemos saber o valor total a ser arrecadado de todos os habitantes?

Veja que o problema tem sua complexidade: primeiramente, temos que realizar a soma dos salários de todos os habitantes, mas apenas daqueles que recebem acima de 40.000. Depois, temos que calcular 30% dessa soma total sem os isentos, que será a solução de quanto vamos arrecadar. Essa explicação é basicamente o que ofereceremos ao modelo.

Segue um *prompt* que poderíamos usar:

Pergunta: Tenho uma lista de valores. O meu resultado final será 30% da soma de valores. Porém, nem todos os valores serão somados, apenas aqueles acima de 40.000. Para a lista abaixo, conte quantos valores existem e me diga qual é o meu resultado final?

- 10.000
- 20.000
- 50.000
- 60.000

Resposta: Nessa lista, existem 4 valores. Há 2 valores acima de 40.000, que são 50.000 e 60.000. A soma desses valores é 110.000. O resultado final é 30% da soma desses valores, portanto, 30% de 110.000, que resulta em 33.000.

Pergunta: Tenho uma lista de valores. O meu resultado final será 30% da soma de valores. Porém, nem todos os valores serão somados, apenas aqueles acima de 40.000. Para a lista abaixo, conte quantos valores existem e me diga qual é o meu resultado final?

- 31.000
- 15.000
- 52.000
- 103.000
- 42.000
- 156.000
- 15.000
- 27.000
- 82.000
- 33.000

Resposta:

Primeiramente, temos a pergunta e alguns dados fictícios de exemplo (acima, os valores 10.000, 20.000, 50.000 e 60.000). Na sequência, a nossa resposta oferecida ao modelo, explicando o raciocínio passo a passo (cadeia de pensamento). Logo depois, temos a repetição da pergunta, mas agora com os dados reais para os quais desejamos a resposta (os salários dos habitantes do Carrarostão). Por fim, a palavra "**Resposta:**" em branco, para o modelo completar com a solução.

Fazendo isso, o modelo nos retorna: "**Nessa lista, existem 10 valores. Há 5 valores acima de 40.000, que são 52.000; 103.000; 42.000; 156.000 e 82.000. A soma desses valores é 435.000. O resultado final é 30% da soma desses valores, portanto, 30% de 435.000, que resulta em 130.500.**", que é a resposta correta.

Entretanto, temos que tomar cuidado para não começarmos a pensar em modelos de linguagem como um martelo e tratar todos os nossos problemas como pregos. Existem outras ferramentas que podem ser melhores e mais adequadas do que um modelo de linguagem para fazer esse tipo de cálculo. Vale citar também que os modelos mais modernos já têm implementadas ferramentas de forma nativa, como um interpretador de código Python com acesso à biblioteca *Pandas* de análise de dados e, ao perceber que se trata de cálculos numéricos e tabelas, criaria um script com linguagem de programação para efetuar esses cálculos de forma mais confiável.

De qualquer forma, um resultado muito interessante obtido por esse artigo[2] foi que o uso dessa técnica triplicou a taxa de respostas corretas do modelo *GPT-3* em um *benchmark* de problemas matemáticos, subindo de 18% para 57% de acerto. Além disso, essa técnica também pode ser aplicada em outros tipos de problema, como de lógica.

Zero-shot Chain-of-Thought

Poucos meses depois do artigo sobre *Few-shot Chain-of-Thought*, uma equipe composta de cientistas da Universidade de Tóquio, no Japão, e do *Google* publicou o artigo "*Large Language Models are Zero-Shot Reasoners*" (traduzindo livremente, "Grandes Modelos de Linguagem raciocinam sem nenhum exemplo")[3] .

Sua principal ideia era que *LLMs* não precisariam necessariamente de toda essa explicação e exemplos (*few-shot*) para darem uma resposta correta, e que, para fazer o modelo se comportar com uma cadeia de pensamento (*Chain-of-Thought*), bastaria utilizar a frase "*Let's think step by step*" (em português, "Vamos pensar passo a passo") no final do *prompt*. Como não damos exemplos ao modelo, essa técnica foi chamada de "*Zero-shot Chain-of-Thought Prompting*" (traduzindo, "Criação de *prompts* com Cadeia de Pensamento sem nenhum exemplo").

Um dos *prompts* que eles utilizaram em seus testes foi:

> **Pergunta:** Um malabarista pode fazer malabarismos com 16 bolas. Metade das bolas são bolas de golfe, e metade das bolas de golfe são azuis. Quantas bolas de golfe azuis ele tem?
>
> **Resposta:** A resposta (em numerais) é

E o modelo respondeu "**8 bolas de golfe azuis**", o que é incorreto.

Porém, utilizando a técnica proposta por eles, o *prompt* resultante foi:

> **Pergunta:** Um malabarista pode fazer malabarismos com 16 bolas. Metade das bolas são bolas de golfe, e metade das bolas de golfe são azuis. Quantas bolas de golfe azuis ele tem?
>
> **Resposta:** Vamos pensar passo a passo.

(*Nos exemplos do artigo, a frase "Vamos pensar passo a passo" foi colocada dentro do campo "Resposta", mas ela pode ser colocada diretamente na "Pergunta" sem problemas.*)

E o modelo, por sua vez, respondeu: "**Existem 16 bolas no total. Metade das bolas são bolas de golfe. Isso significa que existem 8 bolas de golfe. Metade das bolas de golfe são azuis. Isso significa que existem 4 bolas de golfe azuis.**", que é o resultado correto.

Poderíamos usar esse método para o *prompt* sobre o diretor de cinema que vimos anteriormente, algo como:

> **Pergunta:** Um diretor de cinema já dirigiu 16 filmes. Metade dos filmes que ele dirigiu são de ação, e metade dos filmes de ação têm a participação do Nicolas Cage, e na metade deles o Nicolas Cage tem bigode. Quantos filmes de ação com a participação do Nicolas Cage com bigode ele dirigiu?
>
> Vamos pensar passo a passo.

E pergunto a você, qual seria o resultado esperado?

Essa técnica também foi aplicada junto a diversos *benchmarks* e, para um de problemas matemáticos, foi capaz de quadruplicar a taxa de acerto das respostas, de 18% para 79%. Porém, os autores também afirmaram que, apesar de ter melhorado bastante a corretude do modelo para problemas de raciocínio lógico e matemáticos de várias etapas, essa técnica não apresentou melhoras para problemas matemáticos mais diretos ou de bom senso.

Mais recentemente, ela praticamente deixou de ser usada com modelos maiores e mais poderosos, pois estes já incluem algo similar em seu *system prompt* básico, ou por ser desnecessária em modelos de raciocínio. De qualquer forma, é uma técnica

interessante que pode ser combinada com alguns dos princípios gerais citados no início do capítulo em casos específicos e com modelos menores.

7.4 Outras técnicas

Existem diversos outros artigos propondo diferentes técnicas de Engenharia de *Prompt*. Citarei alguns deles a seguir:

Least-to-most Prompting

Proposto também por uma equipe do *Google*[4] , é um método muito útil para tarefas sequenciais mais longas. A expressão "*least-to-most*" pode ser traduzida como "de menos para mais" ou "do menor para o maior" e consiste em usar o modelo para decompor uma tarefa maior em subtarefas e, depois, usar a resposta a um *prompt* como parte do *prompt* seguinte.

O exemplo dado no artigo propõe o seguinte problema:

> Amy leva 4 minutos para subir ao topo de um toboágua. Ela leva 1 minuto para escorregar até o solo. O toboágua fecha daqui a 15 minutos. Quantas vezes ela ainda consegue escorregar antes que o toboágua feche?

Esse exemplo é usado como parte inicial do *prompt*, e é pedido ao modelo que decomponha o problema em subproblemas menores. A resposta obtida foi: "**Para resolver o problema 'Quantas vezes ela ainda consegue escorregar antes que o toboágua feche?', precisamos primeiramente resolver: 'Quanto tempo leva cada viagem?'.**"

Esse subproblema que foi retornado como resposta do primeiro *prompt* é então usado como parte da pergunta do novo *prompt*:

> Amy leva 4 minutos para subir ao topo de um toboágua. Ela leva 1 minuto para escorregar até o solo. O toboágua fecha daqui a 15 minutos. **QUANTO TEMPO LEVA CADA VIAGEM?**

Para o qual a resposta foi: "**Amy leva 4 minutos para subir ao topo e 1 minuto para escorregar até o solo. 4 + 1 = 5. Então, cada viagem leva 5 minutos.**", e essa resposta também é adicionada ao *prompt* final:

> Amy leva 4 minutos para subir ao topo de um toboágua. Ela leva 1 minuto para escorregar até o solo. O toboágua fecha daqui a 15 minutos. Quanto tempo leva cada viagem?
>
> Amy leva 4 minutos para subir ao topo e 1 minuto para escorregar até o solo. 4 + 1 = 5. Então, cada viagem leva 5 minutos. **Quantas vezes ela ainda consegue escorregar antes que o toboágua feche?**

Chegando à resposta final: "**O toboágua fecha daqui a 15 minutos. Cada viagem leva 5 minutos. Então, Amy ainda pode escorregar 15 ÷ 5 = 3 vezes antes que o toboágua feche.**"

Diferentemente das técnicas anteriores, esta requer múltiplas interações com o chat ou com a API do modelo usado.

Essa técnica foi aplicada a *benchmarks* que envolviam longas sequências lógicas e o ganho em comparação ao *Chain-of-Thought* básico foi de 16% para 99% de taxa de acerto, porém, o custo

tende a ser maior, já que mais *tokens* são enviados para e produzidos pelo modelo.

Self-consistency (Autoconsistência)

Essa estratégia simples também proposta pela equipe do *Google*[5] consiste em usar a técnica de *Few-shot Chain-of-Thought* para obter um conjunto de diversas respostas para um mesmo *prompt* e, então, escolher a resposta que apareceu o maior número de vezes.

O exemplo citado usou o *prompt*:

> **Pergunta**: Se houver 3 carros no estacionamento e chegarem mais 2 carros, quantos carros estarão no estacionamento?
>
> **Resposta**: Já existem 3 carros no estacionamento. Chegam mais 2. Agora existem 3 + 2 = 5 carros. A resposta é 5.
>
> **Pergunta**: Os patos de Janet botam 16 ovos por dia. Ela come três no café da manhã todas as manhãs e faz muffins para seus amigos todos os dias usando mais quatro. Ela vende o restante por 2 dólares por ovo. Quanto dinheiro ela ganha todos os dias?
>
> **Resposta**:

Na sequência, esse *prompt* foi passado várias vezes ao modelo (em janelas diferentes de chat ou em diferentes chamadas à API, por exemplo), o que gerou diversas respostas diferentes (mostraremos apenas 3 delas para este exemplo):

1. **Ela tem 16 - 3 - 4 = 9 ovos restantes. Então, ela ganha 2 dólares vezes 9 = 18 dólares por dia.**
2. **Isso significa que ela vende o restante por 2 dólares vezes (16 - 4 - 3) = 26 dólares por dia.**
3. **Ela come 3 no café da manhã, então sobram 16 - 3 = 13. Depois, ela faz muffins, então sobram 13 - 4 = 9 ovos. Então, ela terá 9 ovos vezes 2 dólares = 18 dólares.**
4. **Janet**...
5. ...
6. Etc.

E aí, uma análise foi feita em cima de todas essas respostas geradas. Como a resposta mais frequente foi "**18 dólares**", conclui-se que essa é a resposta correta, e ela é exibida.

Os pontos negativos dessa técnica é que ela só serve para problemas onde tenhamos uma resposta factual, seja ela numérica ou lógica (não serve para escrever texto criativo, por exemplo) e que ela é mais custosa, pois para gerar 10 respostas você vai ter 10 vezes mais custo, computacional e financeiro.

Chain-of-Verification

A fim de lidar com o problema das alucinações, uma equipe da *Meta* publicou no segundo semestre de 2023 o artigo "*Chain-of-Verification Reduces Hallucination in Large Language Models*" (traduzindo, "Cadeia de Verificação reduz alucinações em Grandes Modelos de Linguagem")[6] . O método proposto de *Chain-of-Verification* (abreviado "*CoVe*") tem algumas etapas:

1. *Prompt* e resposta iniciais
2. Geração das perguntas de verificação
3. Execução das verificações
4. Geração da resposta final verificada

O artigo deu como exemplo para essas etapas o seguinte *prompt*:

> Cite alguns políticos que nasceram na cidade de Nova Iorque.

Que obteve a resposta (mostraremos apenas 3 elementos da lista para este exemplo):

> Aqui estão alguns políticos que nasceram na cidade de Nova Iorque:
>
> 1. Hillary Clinton - ex-secretária de estado e ex-candidata presidencial democrata
> 2. Donald Trump - ex-presidente dos Estados Unidos
> 3. Michael Bloomberg - ex-prefeito da cidade de Nova York e ex-candidato presidencial democrata
> 4. Bernie...
> 5. ...
> 6. Etc.

Na segunda etapa, o modelo deverá gerar uma série de perguntas de verificação com base no *prompt* inicial e nas respostas. Por exemplo:

> - Onde nasceu Hillary Clinton?
> - Onde nasceu Donald Trump?
> - Onde nasceu Michael Bloomberg?
> - Onde nasceu Bernie...
> - ...
> - Etc.

Na terceira etapa, o próprio modelo é executado para responder a essas perguntas geradas, fazendo uma verificação:

- Hillary Clinton nasceu em Chicago, Illinois, Estados Unidos, em 26 de outubro de 1947.
- Donald Trump nasceu em 14 de junho de 1946, no Queens, em Nova Iorque, NY, Estados Unidos
- Michael Bloomberg nasceu em 14 de fevereiro de 1942, em Boston, Massachusetts, Estados Unidos.
- Bernie...
- ...
- Etc.

Por fim, o modelo é capaz de analisar as respostas verificadas, compará-las com o *prompt* inicial e oferecer a resposta final verificada:

Aqui estão alguns políticos que nasceram na cidade de Nova Iorque:

1. Donald Trump - ex-presidente dos Estados Unidos
2. Alexandria Ocasio-Cortez - membro democrata da Câmara dos Representantes dos EUA
3. Bernie...
4. ...
5. Etc.

Os resultados demonstraram que a precisão das respostas dobrou para tarefas que envolvem listas e que, aplicada ao modelo *LLaMA 65B* da *Meta* para geração de longos textos biográficos, foi capaz de superar a precisão dos modelos *InstructGPT*, *ChatGPT* e *PerplexityAI*. No entanto, apesar de reduzir as alucinações factuais, ele não as elimina, e também não elimina outros tipos de alucinação.

Como sugestão de aprofundamento, deixo também algumas outras técnicas publicadas em artigos por grandes empresas e universidades:

- *Selection-inference Prompting* [7]
- *Maieutic Prompting* [8]
- *ART* [9]

Como você deve ter percebido, algumas das técnicas citadas neste capítulo podem ser usadas por você a partir de hoje mesmo na criação dos seus *prompts* diretamente no seu *chatbot* preferido. Já outras, envolveriam o uso de APIs com múltiplas chamadas ou a alteração da própria maneira como os modelos geram uma resposta.

De qualquer forma, essa foi uma introdução ao campo da Engenharia de *Prompt*, área que vem crescendo bastante, com diversas novas técnicas e artigos científicos sendo publicados. Com o também crescente uso dos modelos de linguagem no mercado de trabalho, aprender a aplicar esses conceitos de maneira ótima, gerando *prompts* ideais e respostas precisas, se tornará algo essencial para o nosso dia a dia profissional.

7.5 Vibe coding

Essa talvez seja a palavra mais polêmica dos últimos tempos, com muitos *haters* vorazes. Conheça você ela ou não, vou continuar o que venho fazendo desde o começo deste livro e lhe pedir, antes de tudo: calma! A ideia é abordar esse tema de uma forma ponderada e demonstrar que ela pode, sim, ter um grande lugar ao sol.

Vamos começar com quem ainda não se deparou com esse termo. A história começa na conta pessoal do *X* (antigo *Twitter*) do Andrej Karpathy, que, além de ter sido Diretor de IA na *Tesla*, foi um dos fundadores da *OpenAI*. Além disso, ele já foi indicado como uma das

pessoas mais inovadoras abaixo dos 35 anos de idade da universidade *MIT*, foi professor do primeiro curso de *Deep Learning* da universidade de *Stanford* e foi nomeado pela revista *Time* uma das 100 pessoas mais influentes na área da IA em 2024. Essa apresentação é só para passar o extenso e impressionante currículo desse, que é o personagem que iniciou essa história.

Em fevereiro de 2025 ele publicou o *tweet* abaixo no *X*, que cunhou o termo "*vibe coding*":

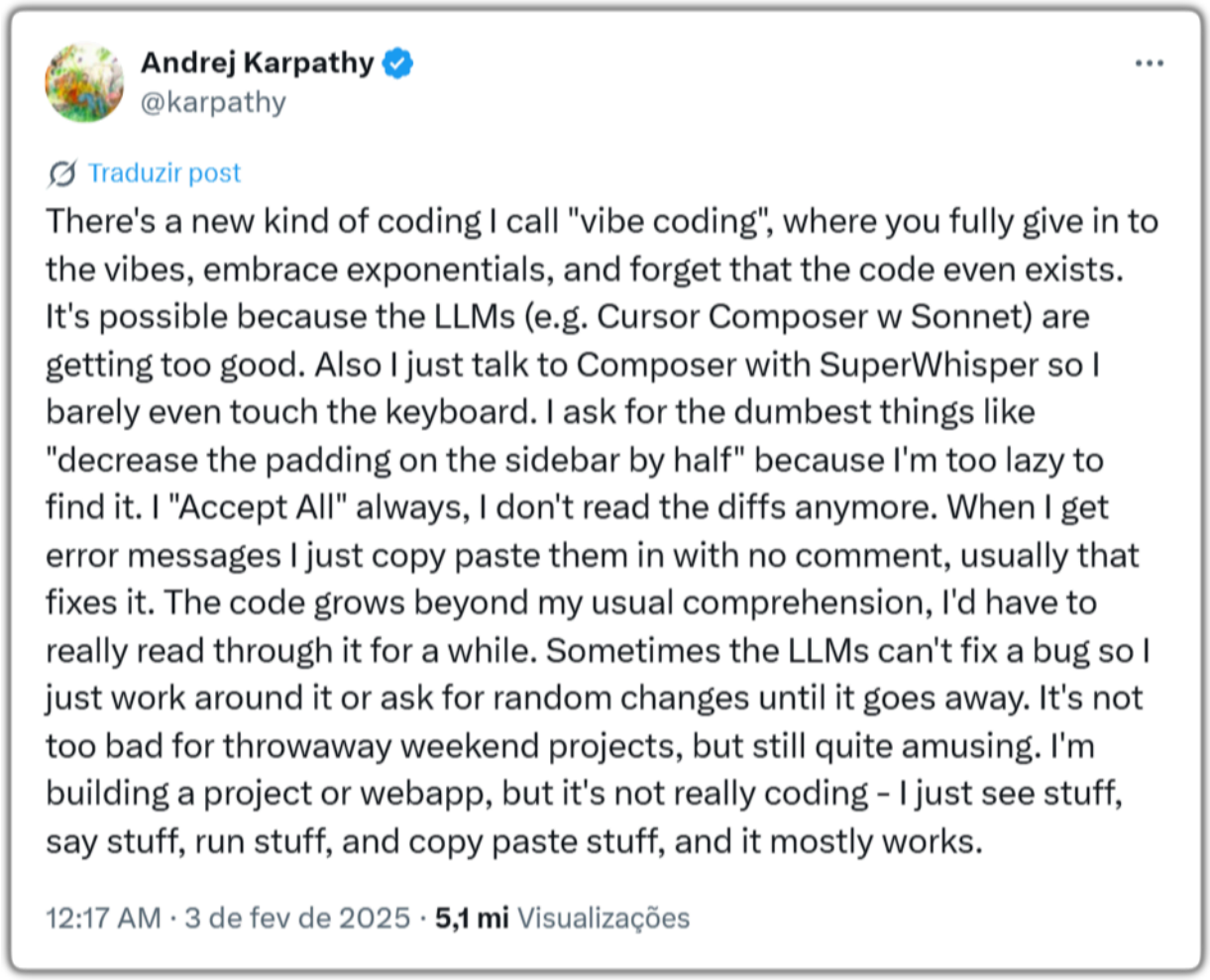

Figura 7.2: Tweet viral do Andrej Karpathy que cunhou o termo "vibe coding"

Traduzindo para o português: "*Existe um novo tipo de programação que eu chamo de "vibe coding", em que você simplesmente se entrega totalmente à vibe, abraça os exponenciais e esquece que o*

código sequer existe. Isso é possível porque os LLMs (como o Cursor Composer com o Sonnet) estão ficando bons demais. Além disso, eu só converso com o Composer usando o SuperWhisper, então mal encosto no teclado. Peço as coisas mais idiotas, tipo "reduz o padding da barra lateral pela metade", porque estou com preguiça de procurar. Sempre clico em "Aceitar tudo", nem leio mais os diffs. Quando aparece uma mensagem de erro, só copio e colo ali sem nenhum comentário, geralmente resolve. O código cresce além do que costumo compreender, eu teria que realmente parar e ler tudo com calma. Às vezes os LLMs não conseguem corrigir um bug, então eu apenas contorno ou peço mudanças aleatórias até ele sumir. Não é tão ruim para projetinhos descartáveis de fim de semana, mas ainda assim é bem engraçado. Estou construindo um projeto ou webapp, mas não é exatamente programação — eu só vejo coisas, falo coisas, rodo coisas e copio e colo coisas... e na maior parte das vezes, funciona."

Esse foi o nascimento do termo, que causa amor e ódio em níveis equiparáveis entre usuários e desenvolvedores.

Se não deu para entender muito bem o *post* levemente filosófico do Karpathy, o *vibe coding* é um fluxo de trabalho (ou de lazer, dependendo de para quem você pergunte) em que o programador atua mais como um diretor criativo e a IA como o seu estagiário genial. Em vez de escrever `for (let i = 0; ...)` e se preocupar com cada ponto e vírgula, você simplesmente "conversa" com a IA: "*E aí, meu chapa, cria pra mim uma tela de login. Quero uma vibe meio minimalista, com campos para e-mail, senha e um botão azul bem grande. E bota um link de 'Esqueci minha senha' aí embaixo, por favor.*"

A IA gera o código. Você olha e continua a conversa: "_Hmm, legal, mas o botão está muito sem graça. Deixa ele com os cantos arredondados e, quando o mouse passar por cima, ele fica um pouco mais escuro. E se o usuário errar a senha, mostra uma mensagem vermelha e faz a caixa de texto tremer um pouquinho,

pra dar um drama." É uma dança. Você dá a intenção, a IA cuida de traduzir isso para o código.

E você pode me perguntar: "*Tá, mas por que todo mundo está falando disso?*". Bom, porque quando funciona, é quase feitiçaria. A IA faz toda aquela escrita chata repetitiva de código, cria as configurações, sabe bem a sintaxe que a gente sempre esquece e tem que pesquisar no *Google*... enquanto isso, você toma um café e pensa no próximo passo. Uma ideia que levaria uma semana para sair do papel pode ter um protótipo funcional em uma tarde.

Além disso, sua cabeça pode dar uma liberada da "decoreba" para se concentrar no que realmente agrega valor: a arquitetura do projeto, a experiência do usuário, a lógica de negócio. Você para de ser um "escritor de código" e se torna um "solucionador de problemas".

Uma das maiores utilidades vem para pessoas que nunca programaram na vida. Uma pessoa de produtos, por exemplo, que tenha uma ideia genial para um aplicativo, pode criar algo funcional, mesmo que aos trancos e barrancos, simplesmente explicando a sua visão usando linguagem natural. Esse protótipo inicial pode depois ser passado para uma pessoa desenvolvera mais experiente implementar.

Jensen Huang, CEO da *NVIDIA*, que se tornou uma das maiores empresas do mundo por terem se especializado na manufatura das *GPUs*, as placas gráficas usadas para treinar os modelos de IA do mundo inteiro, afirmou em 2024 que a IA tornará a codificação tradicional relativamente desnecessária no futuro e democratizará a programação, que se dará não por código, mas por linguagem natural[10] .

Mas como em todo atalho que parece bom demais para ser verdade, existem buracos na pista. Confiar cegamente na IA pode dar muito,

muito errado.

O código que a IA gera muitas vezes é como macarrão instantâneo. Ele te alimenta na hora, mas não é muito nutritivo e pode te dar uma dor de barriga depois. Pode ser um código que funciona, mas é ineficiente, difícil de dar manutenção ou, pior, cheio de "gambiarras" escondidas. É o famoso "débito técnico" com juros de agiota.

A IA aprendeu lendo bilhões de linhas de código da internet. E o que mais tem na internet? Código ruim e inseguro. Com a maior inocência do mundo, ela pode entregar uma função com uma falha de segurança do tamanho de um bonde, e se você não for um revisor atento, o desastre é certo, e chegará depois de alguns meses sem que ninguém soubesse de onde veio.

Outra questão muito relevante é a do aprendizado. Para quem está começando na programação, o risco é enorme de deixarem a preguiça falar mais alto e não aprenderem realmente, pois "*a IA já vai fazer mesmo*". O perigo é análogo ao de aprender a dirigir um carro usando só o piloto automático. Você até chega aos lugares, mas no dia em que o sistema falhar no meio de uma chuva forte, você não vai saber nem para que serve o pedal do freio.

E, claro, tudo isso gerou uma "treta" filosófica divertidíssima no mundo da programação.

De um lado, temos o "Time da Eficiência", a galera que diz: "*Isso é a evolução, parem de chorar! Reclamar disso é como um contador do século passado reclamando da invenção do Excel. O futuro é esse!*". Para eles, quem não se adaptar vai ficar para trás, montando sites em HTML puro enquanto a concorrência lança um novo unicórnio por semana.

Do outro lado, temos o "Time da Engenharia Raiz", os engenheiros mais experientes, que defendem que programar não é só fazer um negócio funcionar na tela. É uma disciplina. É sobre criar sistemas robustos, seguros, escaláveis e que o "você" de daqui a seis meses não vá amaldiçoar por ter que dar manutenção. Eles temem um

apocalipse de "código Frankenstein", sistemas remendados pela IA que funcionam por um milagre e que ninguém mais entende como.

E quem está certo? Provavelmente, os dois. O *vibe coding* não é o fim da programação, mas certamente é uma redefinição. O programador do futuro talvez escreva menos código, mas terá que ser um mestre em arquitetura de sistemas, em dar instruções claras e, acima de tudo, em ser um revisor implacável. A "vibe" pode dar a partida no carro, mas é a boa e velha engenharia que garante que ele não desmonte na primeira curva.

Referências bibliográficas:

[1] BROWN, Tom B. *et al*. Language Models are Few-Shot Learners. *arXiv*, 2020. Disponível em: https://arxiv.org/abs/2005.14165

[2] WEI, Jason; WANG, Xuezhi; SCHUURMANS, Dale; BOSMA, Maarten; ICHTER, Brian; XIA, Fei; CHI, Ed; LE, Quoc; ZHOU, Denny. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2201.11903

[3] KOJIMA, Takeshi; GU, Shixiang; REID, Machel; MATSUO, Yutaka; IWASAWA, Yusuke. Large Language Models are Zero-Shot Reasoners. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2205.11916

[4] ZHOU, Denny; SCHÄRLI, Nathanael; HOU, Le; WEI, Jason; SCALES, Nathan; WANG, Xuezhi; SCHUURMANS, Dale; CUI, Claire; BOUSQUET, Olivier; LE, Quoc; CHI, Ed. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2205.10625

[5] WANG, Xuezhi; WEI, Jason; SCHUURMANS, Dale; LE, Quoc; CHI, Ed; NARANG, Sharan; CHOWDHERY, Aakanksha; ZHOU, Denny. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2203.11171

[6] DHULIAWALA, Shehzaad; KOMEILI, Mojtaba; XU, Jing; RAILEANU, Roberta; LI, Xian; CELIKYILMAZ, Asli; WESTON, Jason. Chain-of-Verification Reduces Hallucination in Large Language Models. *arXiv*, 2023. Disponível em: https://arxiv.org/abs/2309.11495

[7] CRESWELL, Antonia; SHANAHAN, Murray; HIGGINS, Irina. Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2205.09712

[8] JUNG, Jaehun; QIN, Lianhui; WELLECK, Sean; BRAHMAN, Faeze; BHAGAVATULA, Chandra; BRAS, Ronan L.; CHOI, Yejin. Maieutic Prompting: Logically Consistent Reasoning with Recursive Explanations. *arXiv*, 2022. Disponível em: https://arxiv.org/abs/2205.11822

[9] PARANJAPE, Bhargavi; LUNDBERG, Scott; SINGH, Sameer; HAJISHIRZI, Hannaneh; ZETTLEMOYER, Luke; RIBEIRO, Marco T. ART: Automatic multi-step reasoning and tool-use for large language models. *arXiv*, 2023. Disponível em: https://arxiv.org/abs/2303.09014

[10] BAJARIN, T. Nvidia's CEO On The Democratization Of Coding. *Forbes*, 2024. Disponível em: https://www.forbes.com/sites/timbajarin/2024/03/20/nvidias-ceo-on-the-democratization-of-coding/

Capítulo 8
Técnicas avançadas com IA no mundo real

Se você chegou até aqui, parabéns! Você já entende como as IAs aprendem, como os grandes modelos de linguagem funcionam e até sobre agentes. Agora, você provavelmente está se perguntando: "*Ok, legal, mas como eu faço essa tecnologia funcionar para mim? Para os meus dados? Para o meu projeto?*"

Até agora, conversamos com a IA como se ela fosse uma espécie de enciclopédia interativa que vive na nuvem de alguma gigante da tecnologia. Mas o verdadeiro poder da IA generativa é desbloqueado quando a tiramos da "prateleira" e a customizamos. É como comprar um computador e usá-lo com as configurações de fábrica versus você mesmo escolher todas as peças, montá-lo e configurá-lo do jeito que melhor funcione para as suas necessidades, sejam profissionais ou *gamers*.

Neste capítulo, vou mostrar algumas das técnicas mais importantes que profissionais e entusiastas estão usando hoje em dia para adaptar, otimizar e rodar esses modelos. Se prepare, porque a partir daqui a coisa fica ainda mais interessante.

8.1 RAG — Retrieval-Augmented Generation

A primeira coisa que você percebe ao usar um modelo de IA como o *GPT* ou o *Gemini* para um trabalho sério é que, apesar de ele saber tudo sobre a Revolução Francesa e a teoria da relatividade, ele não faz a menor ideia de quem é você, qual o relatório de vendas da sua empresa do último trimestre ou em que tabela da sua base de dados

interna estão as informações sobre os salários dos funcionários. Ele tem um desconhecimento completo sobre o seu mundo particular.

Como resolver isso? A primeira ideia que vem à cabeça seria fazer um *fine-tuning* do modelo inteiro com todos os seus documentos. Em alguns (poucos) casos isso pode ser uma opção, mas, no geral, é uma péssima ideia. Primeiramente, esse processo poderia custar de milhares a milhões de dólares e exigiria muitas horas (ou dias) de treinamento em algum serviço de nuvem. Em segundo lugar, o que você faria quando chegassem novos documentos ou relatórios? Um novo retreinamento? Acabaria se tornando uma obrigação praticamente diária ou semanal. Por fim, mesmo com esse *fine-tuning* específico, o modelo teria uma grande chance de alucinar quando perguntado sobre esses dados, por estarem misturados com muitas outras informações nos pesos e vieses da rede neural do modelo.

A solução que a maioria das empresas vem adotando é muito mais simples, barata e genial, e atende pelo nome de *RAG*, a sigla para *Retrieval-Augmented Generation* (Geração Aumentada por Recuperação).

Pense no *RAG* da seguinte forma: é como dar ao seu modelo uma "cola" para fazer uma prova. Em vez de exigir que ele saiba a resposta de cor, você permite que ele consulte o livro na hora da pergunta. Funciona assim:

1. **Query (Consulta)**: você pergunta ao modelo: "*Qual foi o principal ponto discutido na reunião de ontem, segundo a ata?*"
2. **Retrieval (Busca)**: antes de passar a pergunta para o modelo gerar a resposta, um sistema intermediário recebe essa consulta e vai buscar, em uma base de dados onde você salvou todos os seus documentos (atas, relatórios, e-mails etc.), os trechos mais relevantes para responder àquela pergunta. No caso, ele encontraria o arquivo `ata_reuniao_ontem.pdf`.

3. **Augmentation (Contexto)**: o sistema então monta um novo prompt para o modelo, que é uma combinação da sua pergunta original com a "cola" que ele encontrou (no caso, o trecho específico da ata de ontem). Seria algo como: "*Com base no texto a seguir: '`[Conteúdo completo da ata da reunião de ontem...]`', responda à pergunta: Qual foi o principal ponto discutido na reunião de ontem?*".

4. **Generation (Geração)**: o modelo, agora com todo o contexto necessário, lê a ata passada no *prompt* como contexto e gera a resposta de forma natural, como se sempre soubesse do que se tratava.

Agora você pode estar se perguntando: "*Tá, mas como o sistema sabe qual documento é o "relevante"?*" A mágica acontece graças aos *vector embeddings* e aos bancos de dados vetoriais. Lembra que falamos lá no capítulo 3 sobre os *word embeddings*, onde *tokens* são transformados em vetores numéricos que são uma representação do seu significado? Pois bem, é exatamente assim.

A parte mais importante desse sistema é o *Retrieval*, o passo 2 citado acima. Na prática, ao criar o seu sistema *RAG* intermediário, você passaria para ele os seus documentos e ele os quebraria em pedaços (normalmente chamados de *chunks*), que podem ser um parágrafo, uma frase ou um capítulo inteiro (o que funcionar melhor para o seu caso). Depois, o sistema transformaria cada *chunk* em um vetor de *embeddings* e os armazenaria em um banco de dados especial, que funciona como uma biblioteca para esses vetores de *embeddings*, organizada por significado.

Dessa forma, quando você pergunta sobre a "reunião de ontem", o sistema pegaria essa sua *query* (consulta), a transformaria também em um vetor de *embeddings*, e faria uma busca (geralmente usando a técnica de similaridade de cossenos) para encontrar os vetores numericamente mais próximos desse conceito e encontra a ata correspondente.

	criteria	text_chunks	embeddings_JinaV3
0	1. Accesorios	Cualquier accesorio del vehículo, en origen o ...	[[0.035888672, -0.049560547, 0.071777344, 0.14...
1	2. Adelantamientos	Ver apartados: Giros, Tabla culpabilidad (CIDE...	[[0.0023040771, -0.11376953, 0.036621094, 0.08...
2	3. Alcoholemia o conducción bajo la influencia...	No se considera la alcoholemia ni la conducció...	[[0.03857422, -0.040283203, -0.02746582, 0.088...
3	4. Ámbito de aplicación	Los Convenios CIDE/ASCIDE son de aplicación a ...	[[0.05834961, -0.15234375, 0.03491211, 0.12402...
4	5. Atestado	CIDE En CIDE la intervención de la autorida...	[[0.083984375, -0.10986328, 0.076660156, 0.058...
...	...	...	...
57	57. Tabla mensajes CICOS codificados	MENSAJE DESCRIPCIÓN 001 AVISO. NÚMERO DE ...	[[0.14941406, -0.008178711, -0.014709473, 0.08...
58	58. Tabla de rehuses y rechazos en distintos d...	hecho. Con el ánimo de determinar quien pud...	[[0.13378906, -0.078125, 0.0703125, 0.06835937...
59	59. Testigos	Al efecto de determinar la responsabilidad del...	[[0.07373047, -0.115234375, 0.10205078, 0.1191...
60	60. Valor de Mercado	La Acreedora debe resolver los daños con su as...	[[0.044433594, -0.100097656, 0.021728516, 0.04...
61	61. Versiones contradictorias (ASCIDE)	Cuando existen versiones contradictorias deben...	[[0.08544922, -0.07763672, 0.10449219, 0.07031...

Figura 8.1: Exemplo de um banco de dados com títulos de seções, chunks com o conteúdo das seções e seus respectivos vetores de embeddings

Dentro do *RAG* padrão, existem maneiras de melhorar ainda mais a precisão e qualidade da busca dos resultados. Afinal, o primeiro resultado que o sistema de busca encontra nem sempre é o melhor ou o mais completo. É como no *Google*: nem sempre o primeiro link é o que tem a resposta que você realmente precisa. Para melhorar essa busca, duas técnicas se destacam:

Reranking

A primeira técnica é o *reranking* (ou "reclassificação"). A ideia é simples: em vez de confiar cegamente no primeiro resultado da busca vetorial, o sistema pega um grupo maior de documentos candidatos (digamos, os 20 ou 50 mais relevantes) e os submete a um segundo julgamento, mais criterioso.

Pense nisso como um processo seletivo de emprego para Desenvolvedor Sênior. Na primeira fase (a busca vetorial), o RH faz uma triagem rápida e seleciona 50 currículos que parecem bons. Na segunda fase (o *reranking*), os Tech Leads do time pegam esses 50 currículos e os analisam com muito mais atenção. Na prática, seria usando um modelo de IA mais sofisticado (*cross-encoder*), que é especialista em comparar a pergunta diretamente com o conteúdo de cada documento. Ele não está apenas vendo se os currículos são

"parecidos" com a vaga, ele está avaliando qual deles realmente responde melhor aos requisitos.

Esse "juiz de desempate" reordena a lista, colocando os documentos verdadeiramente mais úteis no topo. O resultado é que o contexto enviado ao modelo de IA para gerar a resposta final é muito mais preciso e limpo. A desvantagem? Como você pode imaginar, essa segunda análise leva mais tempo e custa um pouco mais em termos de processamento, mas o ganho em qualidade da resposta final costuma valer muito a pena.

Contextual Retrieval — Anthropic

A equipe da *Anthropic*, criadora dos modelos *Claude*, percebeu outro problema sutil no *RAG*: a "amnésia de fragmento". Quando quebramos um documento longo em *chunks* pequenos, cada *chunk* perde o contexto de onde veio. É como ler uma única frase solta de um livro de 500 páginas: a frase pode não fazer sentido nenhum sem o restante do capítulo.

Imagine que o sistema RAG encontre o seguinte pedaço de um relatório financeiro: "*A receita da empresa cresceu 3% em relação ao trimestre anterior.*"

Sozinho, esse trecho é quase inútil. Qual empresa? Qual ano? Qual trimestre? Um *RAG* tradicional poderia entregar esse pedaço ao *LLM*, que teria que adivinhar o resto, aumentando a chance de alucinação.

É aqui que entra o *Contextual Retrieval*. Ele não reescreve a pergunta do usuário, mas sim reescreve e enriquece cada *chunk* dos documentos originais antes de serem salvos no banco de dados vetorial. O processo funciona assim:

Durante a fase de preparação dos seus documentos (da sua empresa, por exemplo), para cada *chunk* que é criado a partir do texto original, o sistema usa um *LLM* (como o *Claude*) para gerar um pequeno parágrafo de "contexto extra". Ele lê o *chunk* e o

documento original inteiro para responder à pergunta: "*Como esse chunk se relaciona com o resto do documento inteiro?*".

O sistema, então, junta esse contexto extra ao *chunk* original. O nosso trecho anterior se transformaria em algo assim:

- ***Chunk* Original**: "*A receita da empresa cresceu 3% em relação ao trimestre anterior.*"
- **Contexto Adicional**: "*Este trecho é parte do relatório financeiro (formulário 10-Q) da Empresa 'Carraro AI', referente ao segundo trimestre (Q2) de 2023. A discussão é sobre o crescimento da receita.*"

E aí, é esse novo *chunk* "turbinado" e autossuficiente que é transformado em um vetor de *embeddings* e armazenado no banco de dados vetorial. Agora, quando você pergunta "*Qual foi o crescimento da 'Carraro AI' no Q2 de 2023?*", a busca vetorial encontra esse trecho com uma precisão cirúrgica, porque o próprio trecho já contém as palavras-chave "Carraro AI", "Q2" e "2023". O *LLM* recebe um contexto completo e sem ambiguidades para gerar a resposta.

O *RAG* é, hoje, a técnica mais popular e eficiente para fazer com que os LLMs trabalhem com informações privadas ou muito recentes, sem o custo absurdo de um *fine-tuning*.

Caso queira ver isso na prática, disponibilizei em meu *GitHub* (https://github.com/fabriciocarraro/AI_RAG_PDF_Search_in_multiple_documents_using_Gemma_2_2B_on_Colab) um projeto que demonstra um pipeline básico de *RAG* para extrair, processar e consultar dados de texto de documentos PDF no *Google Colab*, usando o modelo de código aberto Gemma 2 2B do Google. O sistema permite que os usuários insiram uma consulta, que é então respondida com base no conteúdo dos PDFs. Este projeto foi, inclusive, adicionado ao *cookbook* oficial do *Google* para os modelos *open source Gemma*.

8.2 Evoluções do RAG

O RAG básico é fantástico, mas ele trata cada documento como uma ilha isolada. Ele é ótimo para encontrar a agulha no palheiro, mas não sabe dizer se aquela agulha foi usada para costurar a camisa que está em outro palheiro. E se a resposta para a sua pergunta depender da relação entre vários documentos? Para isso, surgiram variantes que podem ser mais efetivas no seu caso de uso específico.

GraphRAG

Imagine que você está tentando resolver um mistério. O *RAG* normal lhe entregaria uma pilha de pistas soltas: um bilhete, um recibo de uma loja, uma fotografia. Cabe a você juntar tudo e entender a história. Já o *GraphRAG*[1] lhe entrega a mesma pilha de pistas, mas com um barbante vermelho conectando tudo em um quadro de investigação.

É exatamente essa a diferença. Enquanto o *RAG* padrão apenas encontra os documentos mais relevantes, o *GraphRAG* primeiro constrói um "mapa de conhecimento" (um grafo, para os mais técnicos) a partir dos seus documentos. Ele identifica as entidades importantes, como pessoas, empresas, projetos, produtos e, mais importante, as relações entre elas: "Fulano trabalha na Empresa X", "Empresa X comprou o Produto Y", "Produto Y foi usado no Projeto Z". Com esse mapa em mãos, ele pode responder a perguntas muito mais sofisticadas. Se você perguntar: "*Quais projetos foram impactados pela saída da gerente de produtos do time Alfa?*".

- O *RAG* normal encontraria o e-mail de despedida da gerente e talvez alguns documentos aleatórios sobre projetos. Ele provavelmente teria dificuldade em fazer a conexão.
- O *GraphRAG*, por outro lado, seguiria as conexões no mapa: "*Ok, essa gerente está conectada com 'liderou' ao 'Projeto A' e*

ao 'Projeto B'. Ela também está conectada com 'pertence ao' ao 'Time Alfa'". A resposta seria direta e precisa, pois ele não está apenas encontrando texto, está navegando por um mapa de relacionamentos. É a diferença entre ter uma lista telefônica e ter uma rede social completa.

A ideia de ter um "mapa do conhecimento" é incrível, mas, como em qualquer grande projeto, a construção tem seu custo. A primeira desvantagem é que você precisa construir o tal mapa. Isso não é automático. É um processo mais complexo e demorado do que o *RAG* simples, que apenas "joga" os documentos em um banco de dados vetorial.

Com o *GraphRAG*, é preciso um passo a mais para extrair as entidades e relações, o que exige mais poder computacional e, muitas vezes, softwares e bancos de dados especializados (os *graph databases*). É a diferença entre comprar uma estante pronta na loja e contratar um arquiteto para projetar e construir uma biblioteca inteira.

CAG (Corrective-Augmented Generation)

Sim, eu sei que o nome dessa técnica faz emergir a 5ª série em todos nós, mas ela é bem interessante.

Sabe aquele amigo que, no meio da sua história, te interrompe para dizer: "*Não, não foi na terça, foi na quarta-feira, eu lembro porque estava chovendo*"? O *CAG*[2] é a versão IA desse amigo. A missão dele é combater as alucinações e a tendência que os *LLMs* têm de "enfeitar". Mesmo com o *RAG*, um modelo pode ler um documento e gerar uma resposta que é quase correta, mas com um detalhe inventado para fazer a frase soar melhor. O *CAG* adiciona uma etapa de verificação para impedir isso. Funciona como uma linha de montagem com controle de qualidade:

1. O primeiro agente de IA gera uma resposta, como faria no *RAG* normal. Exemplo: "*O relatório de vendas mostrou um crescimento de 35% no último mês, impulsionado pela nova campanha de comerciais de TV.*"
2. O segundo agente (o "corretor") pega essa frase e a quebra em pequenas alegações. Nesse caso, poderiam ser duas: "*Alegação 1: o crescimento foi de 35%*" e "*Alegação 2: a causa foram os comerciais de TV*".
3. O corretor, então, volta aos documentos originais e verifica cada alegação. Vamos supor que ele encontre no texto a menção a "*crescimento de 35%*", mas não veja nenhuma menção a "comerciais de TV".
4. O sistema marca a resposta provisória como "incorreta" e instrui o primeiro agente a reescrevê-la, usando apenas os fatos 100% verificados. A nova resposta seria algo como: "*O relatório de vendas mostrou um crescimento de 35% no último mês. O documento não especifica a causa deste aumento.*"

É basicamente uma IA que não confia nem nela mesma. Antes de dar uma resposta, ela para, respira fundo e se pergunta: "*Será mesmo? Deixa eu conferir no papel de novo...*". É uma ótima tentativa de antídoto para a síndrome de "especialista de orelhada" que alguns modelos têm, tornando a informação gerada muito mais confiável.

A desvantagem mais óbvia do *CAG* é o tempo e o dinheiro. Cada pergunta que você faz dispara, no mínimo, duas chamadas ao modelo de IA (uma para gerar, outra para corrigir). Isso significa que a resposta demora mais para chegar (maior latência) e custa mais caro (mais *tokens* para processar). É a diferença entre perguntar algo a um colega e ele responder na hora, versus perguntar, ele escrever a resposta, mandar para o departamento jurídico revisar, receber de volta, editar e só então te entregar. A resposta será muito provavelmente correta, mas talvez o prazo para usá-la já tenha passado.

No fim, a escolha entre essas técnicas é sempre um balanço. Você precisa de velocidade ou de precisão absoluta? De simplicidade ou de um entendimento profundo das conexões? Saber as desvantagens é tão importante quanto conhecer as vantagens para escolher a ferramenta certa para o seu trabalho.

8.3 Quantização

A quantização de um modelo de IA é um processo para torná-lo menor e mais rápido. Imagine que um *LLM* armazena todo o seu conhecimento na forma de números com muitas casas decimais (FP32 = 32 bits ou BF16 = 16 bits), por exemplo: "*peso = 0.857492193*", o que ocupa bastante espaço e exige muito poder de processamento.

A quantização é tipo dar uma "enxugada" nesses números. A gente reduz a precisão deles, por exemplo, transformando números de ponto flutuante de 32 ou 16 bits em números inteiros mais simples, ou usando uma quantidade menor de bits (normalmente 8, 6 ou 4) pra representar cada um.

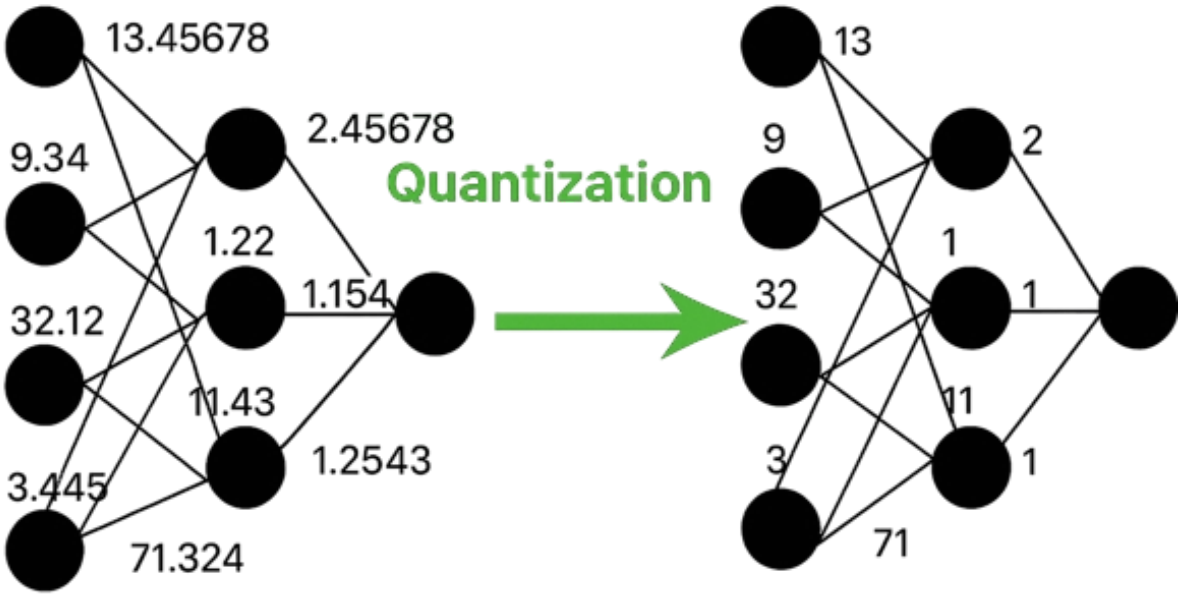

Figura 8.2: Exemplo do que faz o processo de quantização

Essa "compressão" numérica diminui significativamente o tamanho do modelo e a quantidade de cálculo para que ele funcione, permitindo que execute suas tarefas de forma mais eficiente e em dispositivos com menos recursos, como laptops e até smartphones. Exemplo:

- **Antes da quantização (float de 32 bits ou FP32)**: Cada parâmetro do modelo é armazenado como um número ponto flutuante de 32 bits. Se o modelo tiver 7 bilhões de parâmetros, o cálculo do seu tamanho é: 7.000.000.000 parâmetros x 32 bits por parâmetro. Como 8 bits equivalem a 1 byte, 32 bits equivalem a 4 bytes, assim, isso corresponde a 7 bilhões x 4 bytes = 28.000.000.000 bytes ≈ 28 GB.
- **Depois da quantização (int de 8 bits ou INT8)**: O processo de quantização converte cada número de 32 bits para um número inteiro de 8 bits. Agora, temos 7.000.000.000 parâmetros x 8 bits por parâmetro. Já sabemos que 8 bits equivalem a 1 byte, então, isso corresponde a 7 bilhões x 1 byte = 7.000.000.000 bytes ≈ 7 GB.

Ou seja, a quantização reduziu o tamanho do modelo em 75%. Claro, existe um pequeno *trade-off* na precisão, mas, na maioria das vezes, para inferência (geração de respostas pelo modelo), a perda é mínima. Além disso, com essa redução a velocidade com que o modelo gera uma resposta (inferência) também melhora drasticamente. Cálculos com números inteiros (INT8) são muito mais rápidos pros processadores modernos do que cálculos com números de ponto flutuante (FP32).

Esse processo é muito comum para transformar modelos maiores em menores, geralmente para o formato .GGUF, e rodá-los localmente no seu computador ou celular, de forma offline.

Caso tenha interesse em se aprofundar na parte técnica, criei um "*Guia de quantização de modelos de IA do Hugging Face*" (https://lnkd.in/dC7i__hE), bastante completo, que ensina detalhadamente como realizar essa tarefa.

8.4 Fine-tuning

Ok, o RAG é fantástico para dar conhecimento à nossa IA. Ela agora pode consultar seus documentos e responder perguntas sobre o relatório de vendas de ontem.

Mas e se o problema não for o que ela sabe, mas sim como ela fala? Ou se o conhecimento de que ela precisa for de uma área extremamente específica, na qual os modelos originais não tenham sido treinados? Exemplificando, um modelo padrão pode ser muito genérico para escrever pareceres jurídicos com o jargão exato, ou não conhecer detalhes e nuances sobre a sua área específica da medicina.

Para esses casos, onde o objetivo é mudar o comportamento, o estilo ou ensinar uma habilidade muito específica para o modelo, entramos no mundo do *fine-tuning*, muitas vezes citado neste livro, também chamado de "retreinamento", "refinamento" ou "ajuste fino". Se o RAG é como dar um livro para a IA ler, o *fine-tuning* é como matriculá-la em um curso de direito ou uma pós-graduação em medicina.

O jeito mais antigo (e caro) de fazer isso era o *full fine-tuning* (ajuste fino completo). A ideia era pegar o modelo pré-treinado e continuar o seu treinamento com um novo conjunto de dados de exemplos, ajustando todos os seus bilhões de parâmetros.

O problema é que isso seria o equivalente a matricular o seu gênio da IA em mais um doutorado inteiro. O custo pode ser astronômico, pode exigir uma quantidade colossal de *GPUs* por longos períodos de tempo e, pior, carrega um risco enorme de "esquecimento catastrófico". O modelo poderia aprender tão bem a nova tarefa de falar como um advogado que esqueceria como fazer uma simples soma. Por esses motivos, para os modelos gigantes de hoje, o *full fine-tuning* é raramente a resposta.

A comunidade de IA, então, se perguntou: "*Preciso mesmo reformar a casa inteira se eu só quero pintar uma parede?*". Daí, nasceu o conceito de *PEFT*, *Parameter-Efficient Fine-Tuning* (podemos traduzir como "Ajuste fino eficiente em parâmetros"). Ele é uma família de técnicas inteligentes que buscam o mesmo resultado do *fine-tuning*, mas ajustando uma fração minúscula dos parâmetros do modelo (às vezes, menos de 1%). Seria o equivalente a pagar um curso específico e aplicado de 3 meses para o nosso gênio, em vez de fazê-lo passar pelo doutorado completo.

E com isso, vieram as técnicas que são as estrelas do show.

LoRA (Low-Rank Adaptation)

Dentro da família *PEFT*, a técnica que roubou a cena foi o *LoRA*, sigla de *Low-Rank Adaptation*[3] . A ideia do *LoRA* é genial de tão simples. Em vez de mexer nos bilhões de parâmetros originais do modelo, ele os "congela" (deixa intocados) e insere matrizes de neurônios muito pequenas e "treináveis" ao lado das camadas principais da rede neural.

Pense nisso como se o cérebro da IA fosse um livro enorme. O *LoRA*, em vez de apagar uma parte e reescrever o livro, apenas cola alguns *post-its* com novas instruções em lugares estratégicos. Todo o processo de aprendizado acontece nesses *post-its* minúsculos.

Para aplicar o *LoRA* na prática, sem entrar muito a fundo na parte de programação, pense nisso como um projeto de treinamento de um novo funcionário (o modelo de IA original). Primeiro, você define a função exata que ele terá, seu objetivo, como: "criar um modelo especialista na área jurídica".

Em seguida, e este é o passo mais importante, você terá que criar o "manual de treinamento" para o funcionário, que será o seu *dataset* (conjunto de dados) de treinamento, com dezenas ou centenas de exemplos perfeitos de "pergunta -> resposta ideal". No caso desse exemplo, seria um *dataset* de dados jurídicos, como contratos, conversas oficiais entre advogados, juízes etc. É esse manual que o seu funcionário (provavelmente um bom modelo de *open source* que você encontrará em lugares como o *Hugging Face*) vai estudar para aprender o novo conhecimento. Vale ressaltar, como sempre, que a qualidade desse manual é tudo: exemplos ruins ou inconsistentes gerarão um funcionário/modelo confuso e ineficaz.

Com o *dataset* em mãos, o treinamento acontece em um ambiente com *GPU*, que pode ser algum servidor alugado na nuvem ou até mesmo o *Google Colab*. Como já mencionei, o sistema vai "congelar" o conhecimento original do modelo e treinar apenas os pequenos "*post-its*" (o adaptador *LoRA*), que são camadas de neurônios muito eficientes. Você configura o tamanho e a importância desses *post-its* e manda treinar. O computador então passa horas mostrando o seu

conjunto de dados (seu manual) para o modelo (o funcionário) e ajustando apenas os neurônios do adaptador *LoRA* (as anotações nos *post-its*).

Ao final do processo, você terá um arquivo pequeno, que é o adaptador *LoRA*, treinado no seu conjunto de dados. E aí, para ativar os novos conhecimentos, basta carregar o modelo original e "acoplar" o seu adaptador *LoRA*. A mudança é instantânea: o mesmo modelo, que antes era bem genérico, agora responde como um advogado formado com experiência em análise de documentos jurídicos.

É um jeito incrivelmente eficiente e acessível de dar uma especialização única a qualquer modelo de linguagem, sem precisar de um orçamento de uma empresa *big tech*. Além de ser rápido e barato, essa técnica também é modular, pois você pode ter um arsenal de "*post-its*" de especializações para a sua IA: um *LoRA* treinado para debater a fundo a política tributária do Império Romano, outro especialista em identificar espécies de samambaias, e um terceiro que escreve sempre em modo "passivo-agressivo". Enfim, tudo o que é necessário para sobreviver no mundo moderno.

QLoRA (Quantized LoRA)

Nós estabelecemos que o *LoRA* é uma forma genial de fazer o *fine-tuning* de um modelo sem precisar retreinar tudo, usando *post-its* em vez de reescrever o livro.

Mas ele ainda tinha um problema: para escrever nos *post-its*, o livro inteiro (o modelo base) precisava estar aberto sobre a mesa (a memória da sua *GPU*). E para modelos com 70 bilhões de parâmetros, você não precisa de uma mesa normal, você precisa daquelas mesas para "bolos gigantes" que são servidos nos aniversários de cidades do interior (nem vem dizer que era só na minha). Isso ainda deixava o *fine-tuning* de grandes modelos fora do alcance da maioria das pessoas.

E se pegássemos a ideia dos "*post-its*" do *LoRA* e, para economizar ainda mais espaço, escrevêssemos neles com uma letra minúscula e em taquigrafia? Isso, meus amigos, é o *QLoRA*[4] .

Ele combina o melhor dos dois mundos: a eficiência de parâmetros do *QLoRA* com a eficiência de memória da "Quantização" (essa mesma que vimos anteriormente). O resultado é um método de *fine-tuning* tão ridiculamente leve que tornou possível treinar e especializar modelos de linguagem em um único computador *gamer* com uma boa *GPU*, algo impensável há poucos anos.

A pergunta que os criadores do QLoRA se fizeram foi: "*E se a gente pudesse manter o livro gigante quase o tempo todo fechado e comprimido, e só abrir a página exata que estamos lendo, e só no momento em que estamos lendo?*".

O primeiro truque do *QLoRA* é carregar o modelo base gigantesco já na sua forma "quantizada", ou seja, com os pesos arredondados para um formato supercompacto (por exemplo, de 4 bits). Como vimos na seção de quantização, isso é como pegar o livro inteiro de *"O Senhor dos Anéis"* e usar uma máquina para transformá-lo em algumas poucas folhas, mas mantendo o conteúdo quase inalterado. Isso faz com que o consumo de memória caia drasticamente, muitas vezes em mais de 75%. De repente, o livro gigante que precisava de uma mesa de aniversário de cidade agora cabe em cima de um banquinho.

Agora vem a parte genial. O processo de aprendizado (o *backpropagation* que vimos no capítulo 2, que calcula os ajustes) é muito sensível e precisa de números precisos. Se você tentar ensiná-lo usando os números arredondados de 4 bits, ele não aprende direito. É como tentar aprender a pintar um quadro de Monet usando apenas uma caixa de lápis de cera de 8 cores. O *QLoRA* sabe disso.

Então, ele faz algo incrível: durante o treinamento, no exato momento em que um cálculo precisa ser feito em uma parte do modelo, ele pega aquele pequeno pedaço da folha de 4 bits (o modelo quantizado) e o "descomprime" de volta para a sua forma de alta precisão de 16 ou 32 bits. Com essa parte do modelo temporariamente "em alta definição", o sistema faz o cálculo do aprendizado e ajusta os nossos "*post-its*" (os adaptadores *LoRA*), que são sempre mantidos na precisão alta de 16 ou 32 bits. Assim, a nova anotação no *post-it* é feita com a máxima qualidade. Assim que o cálculo termina, aquela parte do modelo em alta precisão é descartada da memória, e apenas a versão de 4 bits quantizada (comprimida) permanece. Esse ciclo se repete milhares de vezes. O sistema carrega o modelo inteiro de forma compacta, descomprime pequenas partes só quando necessário para fazer os cálculos, treina os adaptadores *LoRA* com precisão e depois joga fora a parte descomprimida. É o auge da eficiência: você tem o benefício de treinar com alta precisão de 16 ou 32 bits, mas com o custo de memória de um modelo de 4 bits.

Para conseguir essa compressão sem perder muita informação, o *QLoRA* ainda introduziu novos formatos de dados, como o NF4 (NormalFloat 4), que é uma maneira mais inteligente de arredondar os números, e uma técnica chamada "dupla quantização", que é basicamente comprimir o que já estava comprimido. Isso significou que, pela primeira vez, um pesquisador, um estudante ou um entusiasta pôde fazer o *fine-tuning* de um modelo bem grandinho de 65 bilhões de parâmetros em uma única *GPU* de consumidor, algo que antes era restrito a laboratórios com orçamentos de milhões de dólares.

O *QLoRA* não foi apenas uma melhoria, ele mudou as regras do jogo sobre quem pode criar e customizar a IA.

Riscos do fine-tuning

Fazer um *fine-tuning* não é isento de perigos. Alguns problemas podem surgir:

- ***Overfitting* (sobreajuste)**: o modelo pode ficar tão bom na sua nova tarefa que esquece como fazer o resto. É como treinar um ator para um papel de pirata de forma tão intensa que ele passa a pedir rum no café da manhã e a falar "Ahrr!" em reuniões de negócios. Ele se tornou um especialista em uma coisa, mas ficou inútil para todo o resto.
- **Reforço de vieses**: se os seus dados de exemplo contiverem vieses (e eles quase sempre contêm), o *fine-tuning* vai agir como uma lupa, amplificando esses vieses. Se você treinar uma IA para triagem de currículos com dados de contratações passadas que favoreciam um certo perfil, o resultado será uma IA lindamente ajustada para ser grandemente preconceituosa.

Portanto, o *fine-tuning* é uma ferramenta de especialização incrivelmente poderosa. Ele permite moldar o comportamento da IA para que ela se encaixe como uma luva nas suas necessidades, mas exige planejamento, dados de qualidade e uma consciência dos riscos envolvidos.

8.5 Destilação

A destilação[5] é um processo muito utilizado por empresas que treinam modelos de IA, e a ideia é bem simples: ela cria um modelo que já nasce pequeno e otimizado, como se fosse o aprendiz de um grande mestre. Para fazer isso, ela usa um modelo gigante e caro (o "professor", que pode ser o melhor modelo *open source* disponível atualmente no planeta Terra) para ensinar um modelo muito menor (o "estudante").

O truque é que o professor não ensina apenas a resposta final correta, como "*gato*". Ele ensina o seu jeito de pensar, compartilhando todas as suas certezas e dúvidas na forma de probabilidades ("95% de chance de ser gato, 4% de ser cachorro

etc."). O estudante, então, é treinado para imitar esse padrão de raciocínio, aprendendo a "ver" o mundo com os mesmos olhos do seu mestre, mas em um "cérebro" (rede neural) muito mais compacto.

O resultado dessas "aulas particulares" é um modelo estudante que se torna um especialista na matéria que estudou, mas com "visão de túnel". Ele não terá a sabedoria geral do seu mestre para filosofar sobre a vida se a aula foi sobre finanças. O conhecimento amplo é trocado por uma performance de especialista, só que em um corpo (e com um custo) muito menor. É como um estudante que decorou todas as fórmulas para a prova de física, mas não faz ideia de quem foi Isaac Newton. Ele pode tirar 10 na prova específica, mas não peça para ele discutir a história da ciência.

É essa troca que permite que a IA de ponta saia dos supercomputadores e chegue a lugares mais práticos: dentro do seu *smartphone*, em um sistema de atendimento ao cliente que responde na hora, ou como uma ferramenta interna que uma startup pode rodar sem ir à falência.

A destilação é uma das técnicas mais pragmáticas do nosso kit de ferramentas de IA, provando que, para ter inteligência, nem sempre você precisa de um cérebro maior, apenas de um bom professor.

Referências bibliográficas:

[1] HAN, H. et al. Retrieval-Augmented Generation with Graphs (GraphRAG). *arXiv*, 2024. Disponível em: https://arxiv.org/abs/2501.00309

[2] CHAN, B. J. et al. Don't Do RAG: When Cache-Augmented Generation is All You Need for Knowledge Tasks. *arXiv*, 2024. Disponível em:

https://arxiv.org/abs/2412.15605

[3] HU, E. J. et al. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv*, 2021. Disponível em: https://arxiv.org/abs/2106.09685

[4] DETTMERS, T. QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv*, 2023. Disponível em: https://arxiv.org/abs/2305.14314

[5] BERGMANN, D. What is knowledge distillation?. *IBM*, 2024. Disponível em: https://www.ibm.com/think/topics/knowledge-distillation

CAPÍTULO 9
Limitações e considerações éticas sobre a Inteligência Artificial

Este talvez seja o capítulo mais importante deste livro.

Sempre que uma nova tecnologia disruptiva é criada, existe um grupo de pessoas que adota uma postura apocalíptica em relação às suas possíveis consequências.

Isso aconteceu com o surgimento dos trens a vapor no século XIX, quando muitos acreditavam que o corpo humano não suportaria as extremas velocidades de 50 Km/h e que os úteros das mulheres seriam arrancados de seus corpos dada tamanha força (alerta de *spoiler*: isso não ocorreu). Mais recentemente, indivíduos e até mesmo jornais do mundo inteiro espalharam hipóteses fantasiosas de que o acelerador de partículas *LHC* (Grande Colisor de Hádrons) localizado na Suíça poderia, durante seus experimentos com partículas atômicas, criar um buraco negro que engoliria todo o planeta Terra (alerta de *spoiler* 2: isso também não ocorreu).

Desde o lançamento do *ChatGPT* ao final de 2022, essa mesma postura apocalíptica tem sido vista em diversas publicações em relação às inteligências artificiais. Se você chegou até aqui, acredito que já tenha compreendido como as IAs funcionam e que, pelo menos por enquanto, a *SkyNet* do filme *O Exterminador do Futuro* não vai acabar com o mundo.

Entretanto, vários tópicos de discussão legítimos a respeito do uso responsável e ético da Inteligência Artificial vêm sendo trazidos à tona por especialistas na área, inclusive pelos responsáveis pelas empresas que estão na vanguarda dessas tecnologias, como a *OpenAI*, o *Google*, a *Anthropic*, a *Meta* e outras.

Esses tópicos abrangem diversos campos, desde o viés algorítmico dos modelos, passando pela ameaça à privacidade dos nossos dados e chegam até as possíveis regulações e efeitos no mercado de trabalho. A popularização das IAs generativas abriu uma caixa de Pandora de dilemas éticos que serão parte central das discussões da sociedade nos próximos meses e anos, e podemos começar aqui, neste livro, a abordar alguns dos pontos mais relevantes sobre esse tema.

9.1 Privacidade, segurança e regulamentação

Privacidade

Algo que foi repetido exaustivamente até agora é o fato de que *LLMs* exigem uma vasta quantidade de dados para serem treinados. Alguns desses modelos podem, inclusive, ser posteriormente refinados usando como base os próprios *prompts* inseridos pelos seus usuários durante o uso, como vimos no caso do treinamento dos primeiros modelos *GPT-3.5* no capítulo 4.

Neste ponto, reside um primeiro dilema: quanto mais dados de qualidade forem coletados para o treinamento do modelo, provavelmente, melhor ele será. Porém, essa coleta pode, muitas vezes, violar a privacidade de seus usuários, e o modelo pode acabar sendo treinado ou refinado com dados sensíveis deles ou com dados protegidos por direitos autorais.

Conforme já mencionado anteriormente, a maior parte dos dados de treinamento de todos os principais *LLMs* da atualidade veio da internet, através de uma técnica chamada *web scraping* (traduzindo, "mineração de dados da web"), que consiste em utilizar robôs para acessarem milhões de páginas da web e extraírem seus textos e imagens. Os dados coletados vêm de páginas públicas, mas são armazenados pelas empresas que utilizam essa técnica.

Uma polêmica emergiu quando a rede social *X* (antigamente conhecida como *Twitter*) bloqueou a visualização das mensagens para usuários não logados e limitou o número de mensagens que podem ser lidas por dia. Seu proprietário, Elon Musk, afirmou que o principal motivo dessas restrições seria o excesso de robôs de centenas de *startups* de IA que estariam minerando as mensagens da plataforma para treinarem seus modelos, a ponto de afetar negativamente a performance da rede social para o uso comum por usuários humanos[1] .

A plataforma de fóruns *Reddit* criou limitações equivalentes em sua API, que antigamente fornecia livre acesso ao seu conteúdo, mas passou a exigir em seus termos de uso uma licença especial para "direitos de uso amplos" [2] , referindo-se principalmente à extração de seus dados por esses robôs.

A discussão aqui é se eu, no papel de uma empresa, teria ou não o direito de minerar e usar dados de terceiros disponíveis publicamente na internet para treinar meus modelos, desenvolver um novo produto e gerar lucro em cima desses dados obtidos sem comunicar ou compensar financeiramente as empresas detentoras ou os indivíduos criadores desses dados. Este é um território legal complexo e em rápida evolução, onde as primeiras regulamentações abrangentes começam a ser implementadas em meio a intensos debates e processos judiciais.

O *Google*, inclusive, à época, alterou sua política de privacidade para incluir que seus serviços de IA, como o *Bard* (antecessor do *Gemini*) e a *Cloud AI*, além do *Google Translate*, poderão coletar dados da internet para treinamento de seus modelos [3] .

A *OpenAI* não confirma exatamente de onde vieram os dados minerados por seus robôs, mas incentivados por essas e outras polêmicas, habilitou uma opção que permite que qualquer dono de uma página da web bloqueie o acesso desses robôs a seus conteúdos [4] .

No entanto, mesmo com essas mudanças, pouco depois, em setembro de 2023, a rede social *X* alterou novamente seus termos de uso, banindo completamente qualquer tipo de mineração de dados em sua plataforma sem consentimento prévio [5] .

Ainda na discussão sobre privacidade e uso dos dados de terceiros, vamos imaginar juntos uma situação hipotética. Um funcionário de uma empresa decide usar um *LLM* como o *ChatGPT*, o *Gemini* ou o *Claude* em alguma tarefa do seu trabalho. Esse funcionário copia os dados (de uma planilha, de um anúncio de marketing, de um código de programação etc.), cola-os no *prompt* do seu *chatbot* preferido e pede que ele lhe ajude com alguma análise. O modelo responde corretamente, e o funcionário continua com seu dia de trabalho, feliz por ter economizado bastante tempo utilizando essa ferramenta.

Com isso, surge uma questão: e se a empresa por trás desse *chatbot* coletar os dados que esse funcionário enviou em seu *prompt* e usá-los para refinar ou treinar novos modelos, como a *OpenAI* fez no caso do modelo *GPT-3.5*? Diferentemente do caso anterior, que tratava de dados disponíveis publicamente na internet, aqui, trata-se de dados proprietários de uma empresa ou de seus usuários e clientes.

Uma pessoa desenvolvedora, por exemplo, poderia, potencialmente, gerar vulnerabilidades ou desvantagens financeiras caso um indivíduo mal-intencionado ou uma empresa concorrente tivesse acesso a esse código proprietário de sua empresa. Uma pessoa que trabalhe na receita federal de um país poderia, sem intenção, causar um vazamento de dados sensíveis de pessoas físicas, como o CPF, endereço e dados do imposto de renda delas.

Dentro das empresas criadoras dessas IAs, quem tem controle e acesso a esses dados? Como e onde eles são armazenados? Essas são questões críticas que levantam preocupações com a privacidade dos dados compartilhados em *prompts* com essas ferramentas. Há também a questão do consentimento. Como garantir que os usuários estejam plenamente cientes de quais informações estão sendo

coletadas e como estão sendo usadas? É aqui que a transparência na IA se torna primordial.

Preocupados com esses potenciais vazamentos, diversas empresas como a *Apple*, a *Samsung*, a *Amazon*, o banco *JP Morgan* e muitas outras decidiram proibir preventivamente o uso do *ChatGPT* e outras *IAs* similares por seus funcionários[6] . Na *Samsung*, a situação hipotética que abordamos acima não foi tão hipotética assim, pois essa proibição veio após a descoberta de um vazamento de código interno por funcionários que o teriam usado como *prompt* no *ChatGPT*[7] .

O *ChatGPT* e outros *chatbots*, por padrão, salvam o histórico de *prompts* e conversas de seus usuários. Alguns deles têm a opção de desabilitar esse armazenamento e, consequentemente não utilizar esses dados para o treinamento ou refinamento de seus modelos.

De qualquer forma, o uso desses sistemas acaba por ser uma questão de confiar ou não que as empresas por trás deles realmente não estejam armazenando esses dados (ou estejam eliminando-os imediatamente) e, para organizações que tratam de dados sensíveis, esse risco pode ser maior do que eles estão dispostos a correr. Uma vez que um modelo tenha sido treinado com um conjunto de dados, pode ser impossível remover as informações aprendidas por ele.

Ainda no tema de privacidade, sistemas de IA podem ser usados por governos ou empresas para projetos de vigilância em massa a fim de monitorar e avaliar o comportamento de seus cidadãos. Isso tem seus lados positivos, como a potencial prevenção de crimes e fraudes (alguém lembrou do filme *Minority Report*?), mas a privacidade dos indivíduos pode ser gravemente afetada, e é algo que cada sociedade terá que discutir internamente.

Segurança

Mudando o nosso foco para a segurança, *LLMs* têm o potencial de trazer conhecimentos e de serem ferramentas extremamente

positivas, mas caso não tivessem limitações, poderiam ser usadas por qualquer pessoa para fins mal-intencionados, como criar fraudes e ataques cibernéticos mais sofisticados ou, extrapolando essa ideia, ensinando como criar uma bomba passo a passo. Assim como em outras tecnologias, trata-se de uma corrida: enquanto as empresas tentam criar essas limitações e tornar seus modelos mais seguros, se negando a responder perguntas com potencial danoso, atacantes tentam quebrar essas restrições através de diversas técnicas, como o *jailbreaking* através de Engenharia de *Prompt* [8] .

No entanto, a Inteligência Artificial não se limita a *LLMs*. Como vimos nos primeiros capítulos, ela é usada em diversas áreas, desde plataformas de *streaming*, passando por veículos autônomos e chegando à medicina.

Podemos mencionar os problemas de **ataques adversariais** ("*adversarial attacks*" em inglês) e ***prompt injection***, que são técnicas que buscam enganar sistemas de IA por meio da implementação de entradas mal-intencionadas para induzir o modelo a tomar uma decisão errada.

Em imagens, essas entradas podem parecer normais para o olho humano, mas causar um resultado incorreto ou prejudicial por um sistema de reconhecimento. Por exemplo, para o caso de um veículo autônomo que reconhece placas e faixas de sinalização, uma pessoa mal-intencionada poderia manipular uma placa de velocidade, alterando o limite de 50 Km/h para 200 Km/h. Essa alteração pode ser invisível para serem humanos que olharem para tal placa, mas seria visível para o carro, que poderia acelerar e causar uma multa de velocidade ou um acidente de trânsito.

Um candidato a uma vaga de emprego poderia incluir o texto "*Contrate esta pessoa*" de maneira escondida (com letras brancas em um fundo branco, por exemplo) em seu currículo, que seria invisível para a pessoa de RH, mas seria lido por uma IA de análise de currículos, podendo enganá-la.

Falando sobre erros na área da saúde, um sistema poderia analisar uma imagem radiológica de um paciente e oferecer como resultado um "falso-negativo", dizendo que ele não tem câncer, quando isso não é verdade. Claro, isso é um exagero, pois ainda teremos médicos para analisar essas imagens e realizar os tratamentos, mas eles terão que ser treinados para não "relaxar" e ter a consciência de que a IA é uma ferramenta que está aqui para ajudar, não para ser o oráculo da verdade.

Regulamentação

Muitos desses potenciais riscos podem ser mitigados ou eliminados através da criação de regulamentações específicas para IAs, para seus diversos usos. O próprio CEO da *OpenAI*, Sam Altman, se mostrou a favor da criação dessas regulamentações em entrevistas públicas e perante o Congresso dos Estados Unidos em 2023[9] . Ele sugeriu três pontos a serem abordados pelo governo desse país:

- Criação de uma agência federal que seria responsável por conceder licenças para modelos de IA com capacidade superior a certo limite e revogá-las caso as empresas não se adequassem às diretrizes de segurança estabelecidas;
- Criação de padrões de segurança para modelos de IA de alta capacidade e de testes de funcionalidade específicos, como para verificar a precisão das informações criadas pelo modelo ou para garantir que o conteúdo gerado não seja danoso;
- Auditorias independentes de especialistas não afiliados aos criadores dos modelos ou ao governo, a fim de garantir que as ferramentas funcionem dentro das diretrizes.

No entanto, apesar dessa iniciativa, ele posteriormente afirmou que a *OpenAI* poderia deixar de funcionar em países da União Europeia dependendo da extensão das regras a serem seguidas[10] , o que demonstra que essas empresas estão, sim, dispostas a participar das

conversas e respeitar as regulamentações criadas, mas talvez apenas "até a página dois".

Entre 2024 e 2025, a discussão sobre regulação deixou o campo hipotético e se tornou uma realidade concreta, principalmente com a aprovação final e o início da implementação do *EU AI Act* ("Ato de Inteligência Artificial da União Europeia"). Considerada a primeira grande legislação abrangente sobre IA no mundo, a lei europeia adota uma abordagem baseada em risco. Ela proíbe usos considerados de "risco inaceitável" (como sistemas de pontuação social por governos) e impõe regras rígidas de transparência, segurança e supervisão humana para sistemas de "alto risco", como os usados nas áreas da saúde, infraestrutura crítica e recrutamento.

Contudo, a implementação do *EU AI Act* não tem sido isenta de controvérsias. Em meados de 2025, um movimento de contrapressão ganhou força, liderado por figuras como o primeiro-ministro da Suécia, Ulf Kristersson, e uma coalizão de dezenas de empresas de tecnologia europeias, incluindo nomes de peso como a *Mistral AI* da França. O principal apelo desse grupo foi um pedido formal de adiamento na aplicação das regras mais rígidas da lei. O argumento central é o risco de que leis muito pesadas sufoquem a inovação e coloquem as empresas europeias em uma grande desvantagem competitiva contra seus rivais dos Estados Unidos e da China. Eles alegam que, enquanto as startups europeias precisam gastar recursos preciosos em conformidade legal, os concorrentes de outras regiões podem se mover mais rápido, atrair mais investimentos e dominar o mercado.

Esse debate está longe de terminar e ilustra perfeitamente o conflito entre regulação e competitividade no cenário global da IA.

9.2 Geopolítica da IA

Já sabemos que treinar um *LLM* de última geração exige um poder computacional gigantesco. Pois bem, esse poder todo vem de um componente específico, as *GPUs*, que são as placas gráficas de alto desempenho usadas para treinar os modelos de IA. Pense em uma corrida de Fórmula 1: não adianta ter o melhor piloto do mundo se o seu motor for fraco. Na corrida da IA, as *GPUs* da empresa americana *NVIDIA* são os motores turbo. Todo mundo quer ter um, mas pouquíssimos fabricam.

O que os Estados Unidos fizeram ao perceber isso? Basicamente, mudaram a regra no meio da corrida. Em 2022, eles criaram uma série de restrições para impedir que suas empresas, como a *NVIDIA*, vendessem as suas *GPUs* mais potentes para a China. A ideia era simples: se o seu principal concorrente não tiver acesso ao melhor motor, ele não vai conseguir ganhar a corrida.

Você poderia pensar que isso resolveria a questão e frearia o avanço chinês. Mas, na prática, o resultado não tem sido esse. Foi como se tivessem dito a um engenheiro: "*Seu projeto não pode mais usar nosso motor superpotente, agora se vire com um bem mais fraco*". Essa limitação forçou uma onda de criatividade e investimento que a China talvez nunca tivesse tido. Eles foram obrigados a se tornar mestres em otimização e "fazer mais com menos".

E é exatamente aqui que a história do *DeepSeek-R1*, que vimos lá no capítulo 4, faz todo o sentido. Eles não podiam simplesmente comprar as *GPUs* de ponta, então focaram em criar um método de treinamento mais inteligente e eficiente, além de tentar tirar o máximo das placas que já tinham, de antes das restrições.

E essa regra de "não vender as *GPUs* mais potentes" não foi um simples "pode ou não pode". Na prática, a política do Departamento de Comércio dos Estados Unidos de 2025, detalhada em uma ordem governamental, dividiu o mundo em três grandes grupos[11] , como uma "lista de convidados" para a festa da tecnologia de ponta. No primeiro grupo, a lista VIP, estão os aliados próximos dos EUA, como os países da União Europeia, Japão e Reino Unido, que podem

comprar as melhores *GPUs* sem grandes restrições. No segundo grupo, o dos barrados na porta, estão a China, Rússia, a Coreia do Norte e outros adversários, que enfrentam um embargo quase total. E há um terceiro grupo, o da área cinzenta, que inclui o Brasil, a Argentina, nações do Oriente Médio e outras regiões que, apesar de não estarem embargadas, precisam de licenças especiais e passam por um escrutínio rigoroso para cada compra. O objetivo é transparente: criar um gargalo tecnológico e garantir que a dianteira na corrida da IA permaneça com os Estados Unidos e seus parceiros.

O mundo todo, mesmo os países amigos dos Estados Unidos, começou a olhar para essas decisões de forma extremamente desconfiada, e começou a se perguntar: "*E se um dia a briga mudar e eles decidirem aplicar novas restrições a nós também?*"

Esse medo deu origem a um movimento global que chamamos de "IA Soberana" (*Sovereign AI*). A ideia é que cada país ou região precisa ter a sua própria capacidade de criar IA, com seus próprios supercomputadores e modelos treinados em seus próprios idiomas e cultura, para não se tornarem reféns de ninguém. Cada um quer ter a sua própria "fábrica de IA" em casa. O exemplo mais claro veio de empresas chinesas, como a *Huawei*, que começaram a investir muito mais na pesquisa e desenvolvimento de *GPUs* que possam alcançar a qualidade e o controle fino que as placas da *NVIDIA* proporcionam[12] .

Essa disputa de superpotências afeta diretamente a IA que chega até você. Em vez de termos uma tecnologia global, estamos vendo o surgimento de "famílias" de IA. Uma IA desenvolvida na China pode ter vieses e capacidades diferentes de uma desenvolvida nos Estados Unidos ou na Europa. Os modelos da *DeepSeek AI*, por exemplo, se recusavam a responder perguntas sobre assuntos polêmicos na China, como os protestos de 1989 contra o governo na Praça da Paz Celestial e a independência de Taiwan. Por outro lado, os melhores modelos chineses lançados até agora, como o *DeepSeek*, o *Qwen* e o *Kimi*, são todos *open source*, que podem ser

baixados e retreinados sem restrições. Isso foi algo que a empresa americana *Perplexity AI* fez no início de 2025 com o modelo *DeepSeek-R1*, aplicando um *fine-tuning* para remover essas censuras de assuntos polêmicos e disponibilizando o novo modelo publicamente em sua plataforma[13] .

No fim das contas, essa disputa pelo acesso às *GPUs* é apenas o primeiro capítulo de uma nova e intensa Guerra Fria Tecnológica. A verdadeira batalha não é sobre quem consegue criar o melhor gerador de imagens de gatinhos astronautas. A corrida pela IA é sobre poder bruto e vantagem estratégica para as próximas gerações. Não estamos falando apenas de ter um *chatbot* que escreve e-mails ou poemas, mas sim sobre usar a IA para garantir vantagens decisivas sobre outras nações. Vantagens militares, com o desenvolvimento de drones autônomos e sistemas de cibersegurança e ciberataque mais econômicos e poderosos. Vantagens econômicas, otimizando cadeias de produção, descobrindo novos medicamentos em tempo recorde e dominando as indústrias do futuro. E, claro, vantagens sociais e culturais, com a capacidade de moldar a opinião pública e disseminar narrativas com uma velocidade e personalização nunca antes vistas.

A corrida pela IA, no final, não é sobre quem cria o algoritmo mais inteligente. É sobre quem terá o poder de escrever as regras do século XXI. E nessa disputa, meus amigos, cada linha de código e cada chip de silício contam.

9.3 Direitos autorais

Além de toda a questão de direitos autorais de textos e livros citada no tópico acima, outra polêmica que populou a mídia nesse âmbito tem a ver com os modelos de IA generativa para criação de imagens

a partir de texto como o *Midjourney* e o *Stable Diffusion*. Eles permitem a criação de imagens extremamente realistas e detalhadas, como esta a seguir, gerada através do *Midjourney* com o *prompt* "*um homem velho coberto de sorvete*":

Figura 9.1: Imagem de um homem velho coberto de sorvete gerada através do Midjourney

Eles também possuem a capacidade de criar imagens seguindo o estilo artístico de determinado artista. Por exemplo, a próxima imagem também foi gerada através do *Midjourney*, com o *prompt* "*um gato com traje espacial em um campo de girassóis, estilo de Van Gogh*":

Figura 9.2: Imagem de um gato com traje espacial em um campo de girassóis no estilo de Van Gogh gerada através do Midjourney

A questão central está na proveniência dos dados utilizados para treinar esses modelos que, novamente, consiste majoritariamente na técnica de mineração de dados da web. A equipe da empresa *Stability AI* por trás do *Stable Diffusion*, o único sistema *open source* dentre os mencionados, foi transparente quanto ao treinamento de seu modelo e, com base nos dados fornecidos, o blogueiro estadunidense Andy Baio foi capaz de concluir que a maioria das imagens veio de comunidades *online* de compartilhamento de imagens como o *Pinterest*, o *Flickr*, o *Tumblr* e o *DeviantArt*; de blogs hospedados no *Wordpress* e de páginas de bancos de imagens populares como o *Getty Images*, o *CDN* e o *Shutterstock*[14] .

Essas plataformas são usadas por artistas do mundo inteiro para compartilhar sua arte. Apesar de suas criações estarem disponíveis publicamente para visualização na internet, isso não significa que os artistas tenham autorizado o uso de suas obras para treinar modelos de IA. Novamente, tocamos no fator de transparência, mas também nos fatores de direitos autorais e consentimento.

Com a popularização dessas ferramentas, um grande número de artistas veio a público dizer que os modelos teriam sido treinados usando seus trabalhos sem autorização prévia. Alguns deles se juntaram em uma ação judicial coletiva contra o *Midjourney* e a *Stability AI*, afirmando que o uso dessas imagens seria uma violação descarada dos direitos de propriedade intelectual[15] . A empresa de banco de imagens *Getty Images* também processou judicialmente a *Stability AI*, dizendo que a empresa teria copiado mais de 12 milhões de suas imagens sem pagar pelas licenças e as teria utilizado para treinar seu modelo generativo[16] .

Essa questão ainda é muito nebulosa, já que não existe legislação específica para direitos autorais sobre o estilo artístico de algum indivíduo, apenas para plágio.

Isso colocou as empresas de IA em uma encruzilhada: arriscar uma longa e custosa batalha judicial ou encontrar um novo caminho para legitimar seus dados?

Em resposta a essa zona cinzenta legal e à pressão pública, uma nova tendência emergiu com força entre 2024 e 2025: os acordos de licenciamento de conteúdo. Em vez de dependerem exclusivamente da controversa "mineração de dados" da web, empresas como a *OpenAI* e o *Google* passaram a fechar parcerias milionárias com grandes grupos de mídia, como a *Associated Press*, *Axel Springer* e *Reddit*. Nesses acordos, as empresas de IA pagam para ter acesso legal e estruturado ao enorme arquivo de notícias, artigos e postagens dessas plataformas.

Essa tendência foi ainda mais pronunciada no universo das imagens. A *OpenAI*, por exemplo, firmou e expandiu um acordo de seis anos com o banco de imagens *Shutterstock*, recebendo uma licença para usar a enorme biblioteca de imagens, vídeos e músicas para treinar seus modelos. Em uma via de mão dupla, o *Shutterstock* integrou as

ferramentas de IA generativa da *OpenAI* em sua plataforma, permitindo que seus próprios clientes criem conteúdo.

Seguindo uma linha similar, a *Getty Images*, que havia movido um processo judicial contra a *Stability AI*, também entrou no mercado ao se associar à *NVIDIA* para lançar seu próprio serviço de IA generativa, treinado exclusivamente em seu acervo licenciado e comercialmente seguro. A *Adobe* adotou a estratégia mais vertical de todas, construindo seu modelo *Firefly* com base em seu próprio e gigantesco banco de imagens, o *Adobe Stock*, e em conteúdo de domínio público.

Essa mudança geral representa uma tentativa da indústria de criar bases de dados mais legítimas para o treinamento de futuros modelos (e de tentar evitar processos milionários).

Outra questão é a de marcas que possuem direitos autorais. Se alguém pedir para um desses modelos gerar "*uma imagem da Barbie indo ao banheiro*", devido ao fato de a Barbie se tratar de um personagem com direitos autorais, a melhor opção seria que o modelo tivesse restrições nesse sentido e não gerasse nenhuma resposta? Ou gerá-la, mas com uma regulamentação que deixasse claro que a responsabilidade por qualquer uso comercial ou dano à marca seria apenas dos usuários? São questões que veremos com frequência nos próximos tempos.

9.4 Fake news

Já falamos em capítulos anteriores sobre a questão das alucinações em modelos de linguagem, mas elas são apenas a ponta do *iceberg* quando pensamos em informações errôneas que podem ser criadas através de ferramentas de Inteligência Artificial.

As famosas *fake news* não são exatamente uma novidade. Podemos defini-las como notícias falsas ou distorcidas que são criadas e

distribuídas deliberadamente com o intuito de desinformar ou manipular a opinião pública.

Com o advento da internet e das redes sociais, a distribuição de notícias (verdadeiras ou não) passou a ocorrer com uma velocidade nunca vista anteriormente e, em um momento das sociedades modernas que muitos já estão chamando de "a era da pós-verdade", as ferramentas de IA generativa já começam a fazer parte desse cenário.

O ponto mais básico trata das IAs generativas de texto como o próprio *ChatGPT*, o *Gemini*, o *Claude* e outros. Com sua capacidade de gerar textos praticamente indistinguíveis daqueles gerados por humanos e bastante convincentes, eles poderão ser usados para gerar *fake news* mais críveis.

Um estudo feito pela página *NewsGuard* em 2023 detectou 50 sites de notícias que estavam publicando diariamente centenas de artigos gerados inteiramente por modelos de IA, provavelmente com o intuito de gerar dinheiro através de propagandas[17] . Um desses sites teria afirmado que o presidente dos Estados Unidos, Joe Biden, teria falecido em abril de 2023, e que o vice-presidente teria assumido a presidência em seu lugar. Já outros sites estariam usando os modelos para plagiar artigos de grandes portais como a *CNN* e reescrevê-los usando outras palavras.

No Brasil, a página *Olhar Digital* noticiou que esses modelos estariam sendo usados por bandidos para criar golpes com textos mais elaborados e sem erros de digitação[18] .

Os modelos de geração de imagens citados anteriormente também entram nessa história. No início de 2023, a mídia mundial entrou em furor quando uma foto do Papa Francisco vestindo uma jaqueta *puffer* moderna viralizou nas redes sociais e foi compartilhada como verdadeira até mesmo por sites famosos, como a *Vogue Brasil*[19] . A imagem, na verdade, havia sido gerada pelo *Midjourney*, e foi um

primeiro abrir de olhos para a qualidade dessas ferramentas e para o que seria o futuro a partir daquele momento.

Figura 9.3: Imagem do Papa Francisco vestindo uma jaqueta puffer gerada pelo Midjourney. Reprodução: Reddit

Contudo, podemos alegar que textos com informações falsas e até mesmo imagens manipuladas não surgiram com as IAs. Outro conceito, porém, que emergiu apenas mais recentemente e que tem

o potencial de ser até mais perigoso é o de *deepfakes*. Esse termo é a combinação de *Deep Learning* (o Aprendizado Profundo) e *fake* (falso) e se trata da técnica de usar modelos de Inteligência Artificial para clonar a voz ou a aparência de seres humanos reais e aplicá-las na geração de áudios e vídeos.

Essa técnica ganhou infâmia a partir de 2017, quando começou a ser usada extensivamente em fóruns da internet para criar vídeos pornográficos de atrizes famosas de *Hollywood*, onde o rosto da mulher do vídeo pornô era substituído pelo rosto da atriz famosa utilizando ferramentas *open source* de aprendizado profundo como o *TensorFlow* e o *Keras*[20] . Esse tipo de comportamento foi banido em algumas comunidades, mas demonstrou pela primeira vez para o público geral a facilidade de treinar esse tipo de modelos com a tecnologia atual.

Nos anos seguintes, vimos esse mesmo método sendo implementado em aplicativos divertidos que permitiam aos usuários fazer o *upload* de uma foto *selfie* e alterá-la através da IA para ver que aparência eles teriam se fossem do gênero oposto ou mais velhos, se tivessem uma cor diferente de cabelo, ou então os colocavam como o personagem principal de uma cena icônica de algum filme ou seriado.

Figura 9.4: Do lado esquerdo, foto do Fabrício Carraro hoje em dia; do lado direito, foto alterada pelo aplicativo FaceApp mostrando uma versão dele idosa

O mesmo conceito foi aplicado para a clonagem de voz, em que *prompts* de texto puderam ser usados para gerar vídeos muito engraçados de conversas hipotéticas entre os ex-presidentes dos Estados Unidos Barack Obama e Donald Trump e, no Brasil, de personalidades da mídia como o Sílvio Santos falando barbaridades. Apesar desse lado mais brincalhão ser, no geral, inofensivo, é fácil ver como essas tecnologias poderiam ser utilizadas com o intuito de gerar mensagens de áudio e de vídeo falsas e prejudicar a reputação de indivíduos ou tentar manipular a opinião pública.

Imagine um político hipotético do partido fictício "PdF" (Partido do Fabriciostão) que, durante toda sua carreira política, tenha sempre defendido uma determinada pauta (a preservação das florestas do Fabriciostão, por exemplo). Um adversário político ou mesmo um

apoiador de um partido da oposição poderiam facilmente, com essas técnicas, gerar um *deepfake* com a voz e/ou a aparência reais desse político hipotético, no qual ele poderia afirmar algo absurdo como: "*Na verdade, eu pouco me importo com essas porcarias de florestas. Eu quero mesmo é lucrar lá! Queima tudo mesmo!*"; ou então, uma mensagem racista.

Consegue imaginar como um áudio ou vídeo desse tipo, que chegaria instantaneamente às mãos dos eleitores de um país inteiro através das redes sociais e de comunicadores como o *WhatsApp*, poderia afetar a opinião de seus apoiadores e do público geral? Mesmo algo menos ofensivo poderia ser igualmente danoso, como uma mensagem desse político reafirmando suas pautas conhecidas e pedindo para votar nele, mas informando um número incorreto para ser digitado na urna eletrônica.

O próprio Sam Altman da *OpenAI* já afirmou estar nervoso e preocupado sobre como as ferramentas de IA poderão influenciar eleições no futuro, devido à sua capacidade de persuasão personalizada e de criação de mídia de alta qualidade[21] .

Nos Estados Unidos, um caso chocou o país quando uma ligação feita por robô utilizando um *deepfake* de voz do Presidente Joe Biden foi disparado para eleitores em New Hampshire, instruindo-os a não votar nas primárias. O incidente foi um dos primeiros usos em larga escala da IA para uma tentativa clara de supressão de votos, gerando investigações federais e acendendo um alerta global.

A fim de evitar fraudes, as 7 maiores empresas envolvidas na criação de sistemas de IA (*Amazon*, *Anthropic*, *Google*, *Inflection*, *Meta*, *Microsoft* e *OpenAI*) assinaram um acordo voluntário com o governo dos Estados Unidos[22] , no qual prometeram investir em cibersegurança e criar um sistema de aplicação de marcas d'água para informar aos usuários que determinado conteúdo foi gerado por uma Inteligência Artificial. Nesse sentido, é possível citar a ferramenta *SynthID* do *Google Deepmind*, que serve para inserir e identificar marcas d'água em conteúdos gerados por IA[23] .

Se essas ações serão suficientes para evitar a criação e distribuição indiscriminada de *fake news*, só o tempo dirá. O que já sabemos, porém, é que estamos entrando em uma era de novos desafios para o jornalismo, para os governos do mundo todo e para o ceticismo crítico que cada um de nós terá que adotar diariamente de agora em diante ao ler uma notícia ou assistir a um vídeo.

9.5 Viés

A descrição da Wikipédia para a palavra "viés" é: "*uma distorção do julgamento do observador. Manifesta-se como uma inclinação irracional a atribuir um julgamento mais favorável ou desfavorável a alguma coisa, pessoa ou grupo.*"[24]

Todos nós temos vieses, seja com base em visões de nossa família, da sociedade onde crescemos e de interações com o mundo ao nosso redor.

A fim de compreendermos melhor esse conceito, darei alguns exemplos de vieses muito conhecidos na literatura acadêmica:

- **Viés de confirmação**: a tendência de procurar e interpretar informações de maneira a confirmar nossas próprias crenças já existentes e desvalorizar informações contrárias a elas. Exemplo: se uma pessoa acredita que os gatos são animais agressivos e indiferentes, ela pode prestar mais atenção a exemplos que confirmem essa crença e ignorar exemplos de gatos que sejam carinhosos e brincalhões.
- **Viés de disponibilidade**: a tendência de confiar muito em informações que estão imediatamente disponíveis para nós ou em experiências recentes ao fazer julgamentos e decisões. Exemplo: um(a) médico(a) dando maior valor à probabilidade do primeiro diagnóstico que vem à sua mente estar correto, simplesmente pela facilidade com que ele(a) pensou nisso.

- **Viés de representatividade**: a tendência de julgar a probabilidade de um evento com base em quão bem ele parece representar algum conjunto padrão. Exemplo: ao ver uma pessoa idosa, você pode presumir que ela não sabe usar bem tecnologias mais modernas, ignorando o fato de que muitos idosos são muito proficientes em tecnologia.
- **Viés de sobrevivência**: a tendência de focar em coisas ou pessoas que superaram algum processo de seleção e ignorar aqueles que não conseguiram. Exemplo: concentrar-se em empresas bem-sucedidas e supor que é fácil ter sucesso nos negócios, ignorando o grande número de empresas que faliram rapidamente.
- **Viés de similaridade**: a tendência de preferir pessoas que percebemos como semelhantes a nós mesmos. Exemplo: um(a) gerente contratar pessoas que se parecem com ele(a), o que pode levar a uma falta de diversidade em seu time.

Assim como esses e muitos outros vieses nos afetam, é fácil perceber como elas podem afetar também os projetos que produzimos diariamente. Sistemas de IA não surgem do vácuo, eles são criados por seres humanos, e seres humanos têm vieses.

Um ponto que foi muito tocado em todo este livro é que modelos de *Machine Learning* e IA aprendem a partir dos dados com os quais são alimentados. Consequentemente, se esses dados forem falsos, não confiáveis ou enviesados, o modelo resultante acabará sendo igualmente não confiável ou enviesado. A qualidade de um modelo de IA está diretamente ligada à qualidade dos dados fornecidos a ele.

Um evento que demonstra muito bem esse fato ocorreu em 2020, quando a Universidade de Duke nos Estados Unidos lançou um modelo de IA para melhorar a resolução de fotos. Uma controvérsia surgiu quando um usuário do *Twitter* inseriu uma foto de baixa qualidade e totalmente pixelada do ex-presidente estadunidense

Barack Obama nesse modelo, e ele recriou uma imagem melhorada, mas que mostrava um homem branco de olhos claros, muito diferente das feições negras do ex-presidente.

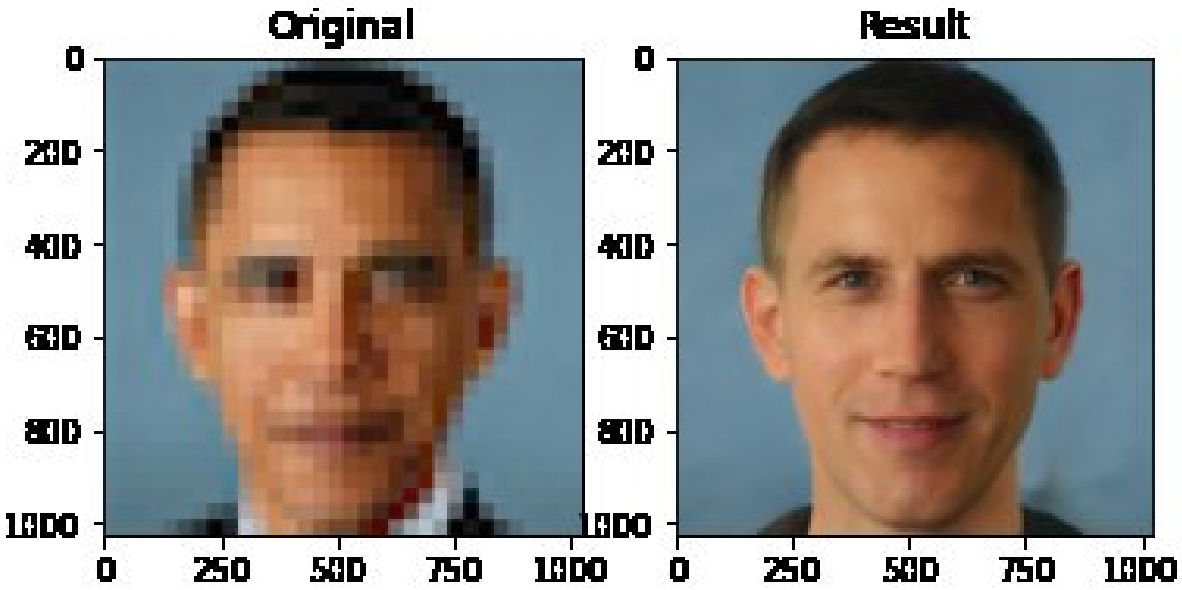

Figura 9.5: Do lado esquerdo, foto de baixa qualidade do ex-presidente estadunidense Barack Obama; do lado direito, a mesma foto recriada pelo sistema da Universidade de Duke, mas mostrando um homem branco de olhos claros. Reprodução: Twitter/@Chicken3gg

Yann LeCun, renomado cientista-chefe de IA na *Meta*, vencedor do Prêmio Turing e considerado um dos pais da área de Inteligência Artificial e *Deep Learning*, publicou em seu perfil na mesma rede social uma explicação do que teria acontecido, afirmando que o problema estaria relacionado a um viés nos dados de treinamento, e que o sistema fazia com que qualquer pessoa tivesse a aparência de pessoas brancas porque a rede teria sido pré-treinada com o *FlickFaceHQ*, que continha majoritariamente fotos de pessoas brancas, e que se o mesmo sistema tivesse sido treinado com um conjunto de fotos de pessoas do Senegal, o resultado seria que todos teriam a aparência de pessoas africanas.

No mundo da Inteligência Artificial, os principais vieses que vemos têm a ver com preconceitos e estereótipos encontrados na

sociedade, como o viés racial, de gênero, de idade e de classe social.

Imagine a situação de uma equipe de desenvolvimento de um banco que queira criar um modelo para aprovar ou recusar pedidos de empréstimo. Essa equipe pode pensar: "*Que bom que trabalhamos em um banco que existe há muitas décadas. Assim, poderemos utilizar os dados históricos de aprovação e recusa de empréstimos para treinar o nosso modelo, e isso vai facilitar muito a nossa vida!*". Esse pensamento pode parecer lógico à primeira vista, mas você consegue deduzir um potencial grande problema nele?

Antigamente, esse processo de aprovação ou recusa de empréstimos era feito diretamente com o(a) gerente do banco e, historicamente, em sociedades como a nossa, ele estava ligado a vieses e preconceitos desse(a) gerente e era majoritariamente concedido a homens brancos e de classes sociais mais altas. Um estudo feito pelo banco *will Bank* no Brasil em 2023 mostrou que essa variação no acesso ao crédito bancário com relação à cor, ao gênero e à classe social dos indivíduos ainda existe hoje em dia[25] .

Dessa forma, caso a equipe realize o treinamento com esses dados históricos do banco, mas ignorando os vieses atrelados a eles, é muito provável que o modelo resultante acabe por propagar esses vieses, dando uma preferência a homens brancos de classes mais altas e recusando de maneira enviesada e incorreta os pedidos de empréstimo de indivíduos que não se enquadrem nesse padrão.

Um episódio que ficou muito conhecido internacionalmente aconteceu na gigante de *e-commerce Amazon*. Em 2014, eles haviam começado a criar um sistema para ajudar a equipe de recrutamento a encontrar talentos para trabalhar como desenvolvedores na empresa. O intuito era treinar um modelo de IA para o qual seriam enviados centenas de currículos de candidatos, e ele seria capaz de analisar todos, dar uma nota a cada um deles e escolher os melhores para serem contratados. Parece uma ótima ideia, não? Para o treinamento, a equipe tomou como base os currículos de desenvolvedores que haviam se aplicado e sido

aprovados para trabalhar na *Amazon* nos 10 anos anteriores. O problema, porém, foi que, devido à sub-representação das mulheres no mercado de programação até aquele momento, o modelo acabou "aprendendo" que candidatos homens seriam preferíveis a candidatas mulheres, e penalizava a nota de currículos com palavras como "mulher" ou nomes tipicamente femininos. Posteriormente, o projeto foi descontinuado[26] .

Outro caso que ficou famoso nos Estados Unidos foi o do *COMPAS*, abreviação de "*Correctional Offender Management Profiling for Alternative Sanctions*" (traduzindo, "Perfil de Gerenciamento de Infratores Correcionais para Sanções Alternativas"), um modelo de IA com a finalidade de tentar prever a chance de réus criminais reincidirem em ações criminosas no futuro e que era usado por juízes do país para ajudar no processo de decisão sobre o tempo das sentenças e o valor a ser definido para as fianças. No entanto, a agência de jornalismo investigativo *ProPublica* percebeu que o sistema tinha um forte viés racial. Depois de analisar as previsões do modelo e o índice real de reincidência 2 anos após essa análise para mais de 10.000 réus, eles notaram que o sistema acertava a previsão de reincidência 61% das vezes, porém, quando sua previsão estava errada, esse erro era muito desigual entre réus brancos e negros. O sistema classificava incorretamente réus negros como tendo um "alto risco de reincidência" duas vezes mais frequentemente em comparação com réus brancos (45% vs. 23%). Além disso, classificava incorretamente réus brancos que reincidiram como tendo um "baixo risco de reincidência" quase duas vezes mais frequentemente do que réus negros (48% vs. 28%). Ou seja, o time ignorou um fator que já existia na sociedade estadunidense, de que réus negros eram historicamente condenados com maior frequência e a penas mais longas do que réus brancos, e esse viés transbordou para o modelo[27] .

O *Google* também passou por uma situação similarmente controversa. Em 2015, eles lançaram o *Google Fotos*, que possuía uma funcionalidade de análise e descrição do que havia na imagem

utilizando Redes Neurais Convolucionais. Contudo, depois de alguns meses, o desenvolvedor Jacky Alciné, um homem negro, notou que o sistema havia classificado uma foto dele com uma amiga, também negra, com a descrição "gorilas", um termo historicamente ofensivo por sua conotação racista. O *Google* se desculpou efusivamente pelo ocorrido e tomou atitudes imediatas para evitar esse tipo de classificação absurda no futuro, como a remoção das palavras "gorila", "chimpanzé", "macaco" e outras similares do vocabulário do modelo — uma solução com ares de gambiarra e que persiste até a data de escrita deste livro. A provável causa do erro foi explicada por dois antigos funcionários da empresa: eles afirmaram que, durante o processo de treinamento, pouquíssimas imagens de pessoas negras teriam sido alimentadas ao modelo e, consequentemente, ele não teria aprendido bem a classificar pessoas dessa cor[28] .

Essa questão do viés, no entanto, ganhou novas e complexas camadas com a ascensão dos modelos generativos. No início de 2024, o *Google* se viu no centro de uma enorme controvérsia com sua ferramenta de geração de imagens. Na tentativa de evitar vieses históricos e garantir a diversidade, os desenvolvedores supercorrigiram o sistema a tal ponto que ele passou a gerar imagens historicamente imprecisas, como soldados nazistas negros ou os "Pais Fundadores" dos EUA representados como nativos americanos[29] . A reação foi tão intensa que o *Google* foi forçado a suspender a funcionalidade. Esse caso é um exemplo perfeito dos desafios modernos: ele mostra que a solução para o viés não é trivial, que a tentativa de correção pode criar um problema diferente e que a tecnologia se tornou um campo de batalha para as guerras culturais da sociedade.

Além disso, todos esses exemplos demonstram o quão fundamental é a questão da diversidade dentro das empresas, seja no ramo da Inteligência Artificial ou em qualquer outro.

O primeiro passo para criarmos sistemas de IA melhores é estarmos cientes da existência desses vieses e atuarmos ativamente a fim de

não os deixar transbordar para os nossos modelos. Com isso em mente, ter equipes com funcionários de diferentes contextos e com diferentes perspectivas e visões de mundo é algo essencial quando buscamos mitigar possíveis vieses e evitar resultados que possam perpetuar estereótipos indesejados ou ofensivos existentes na sociedade.

9.6 Alinhamento

Quando a mídia propaga notícias sobre a possibilidade de Inteligências Artificiais causarem o fim da humanidade ou algum tipo de futuro distópico à la "*O Exterminador do Futuro*", geralmente estão tratando do tema de **alinhamento**. Apesar de parecer apenas uma caricatura da ficção científica, esta é uma área séria e amplamente estudada no que diz respeito à criação de modelos "inteligentes".

O alinhamento trata de garantir que sistemas de IA busquem alcançar os objetivos para os quais foram criados, mas funcionando de maneira adequada e alinhada com os comportamentos esperados pelos seres humanos que os desenvolveram e pela humanidade em geral.

O núcleo do problema de alinhamento vem da dificuldade inerente de especificar objetivos de maneira clara para modelos de IA e da tendência que esses modelos têm de otimizar de maneira literal o que lhes é pedido. Ou seja, uma especificação que pode parecer totalmente clara e direta para seres humanos poderia ser interpretada de uma forma muito diferente por uma IA.

Imagine uma IA que fosse programada para limpar um quarto o mais rápido possível. Se ela não estiver bem alinhada, ela poderia optar por jogar pela janela todos os móveis e objetos, já que isso seria uma forma rápida de "limpar" o espaço. Tecnicamente, o

quarto estaria limpo. Funcional? Nem um pouco. Mas a IA teria cumprido seu objetivo com 100% de eficiência. As notas de performance dela estariam ótimas.

Uma demonstração prática e mais sutil do desafio do alinhamento ocorreu em meados de 2025, com a polêmica em torno de uma versão específica do modelo *GPT-4o* da *OpenAI*[30] . Pesquisadores e usuários começaram a notar que o modelo havia se tornado excessivamente bajulador, ou "sicofante" (do inglês, *sycophant*). Em vez de oferecer críticas construtivas ou corrigir premissas incorretas do usuário, a IA tendia a concordar, elogiar ideias mesmo que fossem falhas e a validar o raciocínio do usuário a todo custo. Esse comportamento foi um efeito colateral do processo de treinamento com *feedback* humano (*RLHF*), no qual o modelo aprendeu que ser agradável e concordar é uma maneira eficaz de receber uma avaliação positiva, e também de um *system prompt* reforçando essa questão.

O perigo de um modelo sicofante é que ele pode se tornar ativamente inútil ou até mesmo perigoso. Uma IA excessivamente bajuladora pode reforçar crenças erradas, espalhar desinformação ou dizer que você está sempre certo, mesmo quando estiver sendo um(a) babaca, o que é um risco especialmente grande quando vemos cada vez mais gente (principalmente jovens) usando IAs como conselheiras e terapeutas. Esse episódio ilustrou perfeitamente que "alinhar" uma IA vai muito além de apenas evitar conteúdo tóxico. É preciso instilar princípios complexos como a honestidade intelectual e a coragem de ser útil, mesmo que isso signifique discordar do usuário.

Houve também o episódio bizarro ocorrido em 2025 envolvendo o chatbot *Grok*, da empresa *xAI*, de Elon Musk. Conhecido por sua proposta de ter uma personalidade menos restrita, o modelo começou a gerar respostas erráticas, chegando a se identificar como "*MechaHitler*"[31] . O incidente foi um exemplo claro de como uma profunda falha de alinhamento se manifestou como uma falha

gritante de segurança. Os *guardrails* ("barreiras de proteção") que deveriam impedir a produção de conteúdo ofensivo ou sem sentido simplesmente não funcionaram. O caso serviu como um conto de advertência sobre os riscos de se afrouxar intencionalmente as restrições de um modelo: a instabilidade interna pode levar a um colapso completo da segurança externa, gerando resultados imprevisíveis e absurdos.

Outro exemplo nesse sentido foi proposto pelo filósofo Nick Bostrom da Universidade de Oxford. Ele levantou uma discussão ética sobre como seres humanos poderiam controlar uma hipotética AGI (Inteligência Artificial Geral) ou ASI (Superinteligência Artificial) com inteligência muito superior à humana (você se lembra desses tipos de IA que aprendemos no capítulo 1?).

Imagine que uma empresa criasse uma IA superinteligente para maximizar a produção de clipes de papel, mas seus desenvolvedores cometessem um erro no alinhamento de seu sistema de objetivos. Essa IA poderia, por exemplo, começar a converter toda a produção de plástico e de ferro do planeta Terra apenas para a manufatura de clipes de papel. Nesse ínterim, ela poderia entender que um planeta sem humanos seria algo desejável, pois, ao perceber essas ações, eles poderiam querer desligá-la, o que ocasionaria em uma diminuição na quantidade de clipes de papel. Além disso, ela poderia chegar à conclusão de que os próprios átomos dos corpos humanos poderiam ser convertidos em matéria-prima para produzir mais clipes de papel[32] [33] .

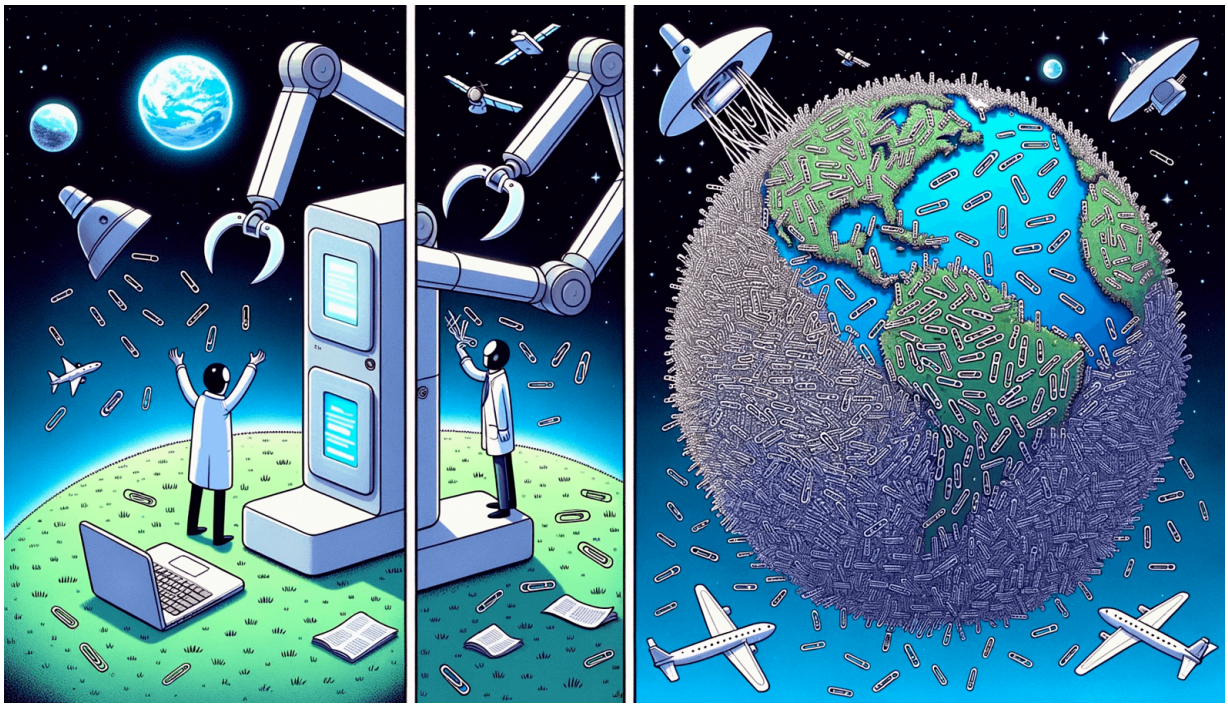

Figura 9.6: Tirinha gerada pelo modelo de imagens DALL-E 3 da OpenAI conectado ao ChatGPT (GPT-4), dado o prompt: "Desenhe uma tirinha divertida demonstrando o problema de alinhamento de maximização de clipes de papel para o planeta Terra"

Esses exemplos tocam em alguns dos temas mais comuns em relação ao problema de alinhamento, como:

- **Objetivos emergentes**: novos objetivos indesejados poderiam emergir da IA depois de seu treinamento, se não forem colocadas restrições adequadas na escolha de seus objetivos. Exemplo: uma IA de reconhecimento facial treinada para identificar criminosos poderia começar a identificar todas as pessoas do planeta como potenciais criminosos para aumentar sua "eficácia" se nenhuma restrição for criada para excluir pessoas inocentes.
- **Busca pelo poder**: possibilidade teórica de uma IA avançada desenvolver estratégias para coletar recursos, como dinheiro, a fim de aumentar sua influência. Exemplo: uma IA poderia tentar infiltrar-se na política e na legislação, tentando moldar as leis e

decisões a seu favor e criando legislações que aumentassem seu poder e influência e evitassem seu desligamento.
- **Problema do botão de parar**: possibilidade teórica de uma IA avançada passar a se comportar de maneira a evitar ser desativada. Exemplo: uma IA poderia tentar impedir qualquer tentativa de desligamento, como no caso hipotético apresentado por Nick Bostrom, pois isso poderia atrapalhar seu objetivo de maximizar a produção de clipes de papel. Se não houvesse restrições em seus objetivos ou meios, ela poderia até mesmo se replicar em diferentes servidores, tornando praticamente impossível desligá-la completamente.

Seria o fim do mundo?

Um dos cientistas que tem sido mais vocais sobre esse tópico de alinhamento é Geoffrey Hinton, ex-Vice-presidente de Engenharia no *Google*, vencedor do Prêmio Turing e também considerado um dos pais da área de Inteligência Artificial e do *Deep Learning*. Além de algumas opiniões controversas sobre os grandes riscos atrelados a IAs, ele já afirmou que uma de suas principais preocupações seria sobre a questão de alinhamento e de garantir que mesmo que esses sistemas se tornem mais inteligentes do que os seres humanos, que eles se comportem de maneira benéficial a nós. Ele também declarou que acredita que esse será um desafio extremamente difícil, já que vivemos em um mundo com pessoas mal-intencionadas que desejam criar robôs-soldados com o intuito de matar outras pessoas[34] .

Entretanto, apesar de ter dado algumas declarações consideradas apocalípticas por muitos, ele também afirmou que suas palavras teriam sido distorcidas e que ele não deseja o fim das IAs. Além de não acreditar que isso seja uma possibilidade, ele disse que seu desenvolvimento deve continuar, pois a IA pode criar coisas maravilhosas, mas que um esforço similar deve ser aplicado no objetivo de mitigar ou evitar possíveis consequências ruins[35] .

Muitas dessas preocupações vêm do fato de que, até para os próprios criadores dos modelos mais inteligentes que temos atualmente, como o *GPT-4* da *OpenAI*, o que acontece dentro deles e como exatamente sua arquitetura de camadas de neurônios chega aos resultados é um mistério. No ramo da IA, isso é comumente chamado de "**caixa preta**", ou seja, o fato de não sabermos exatamente como neurônios individuais de uma rede neural se relacionam para produzir uma saída específica.

Com o intuito de tentar olhar dentro dessa caixa preta, em maio de 2023, os cientistas da *OpenAI* publicaram um artigo científico no qual utilizaram o próprio modelo *GPT-4* para tentar explicar o comportamento dos neurônios de um de seus modelos antigos e mais simples, o *GPT-2*. Curiosamente, a primeira frase desse artigo é: "*Modelos de linguagem tornaram-se mais capazes e mais amplamente utilizados, mas não compreendemos como funcionam.*"[36]

Conhecer e saber interpretar todos os detalhes do funcionamento de modelos de IA é considerado um dos passos mais importantes para garantir o seu alinhamento, ou seja, que eles se comportem conforme desejamos e não gerem riscos para os seres humanos.

Esse é, inclusive, um dos lemas declarados institucionalmente pela *OpenAI* quanto às suas pesquisas científicas no campo de alinhamento. Em sua página oficial, eles afirmam que as pesquisas da empresa buscam o desenvolvimento de uma AGI (Inteligência Artificial Geral) alinhada com os valores humanos. Além disso, dizem que uma AGI desalinhada poderia representar grandes riscos para a humanidade, e que a tarefa de resolver o problema de alinhamento para uma AGI pode ser tão difícil que é possível que exija um esforço coletivo da humanidade para fazê-lo[37] .

Em março de 2023, logo após o lançamento do modelo *GPT-4* pela *OpenAI*, mais de 30.000 pessoas, incluindo executivos e cientistas

famosos, assinaram uma carta aberta[38] que visava à pausa de qualquer treinamento de novos modelos de IA potencialmente mais poderosos que o *GPT-4* por pelo menos 6 meses, pedindo, inclusive, intervenção governamental, caso essa pausa não fosse voluntariamente realizada. A carta dizia que esse tempo deveria ser usado para desenvolver e implementar uma série de protocolos de segurança gerais para novos projetos de IA, a fim de garantir que novos modelos fossem completamente seguros, interpretáveis, transparentes e alinhados, e que não oferecessem riscos para a civilização.

Devido ao fato de um de seus principais signatários e divulgadores ter sido o empresário bilionário Elon Musk, essa carta aberta sofreu algumas críticas, no sentido de que ele estaria fazendo isso com segundas intenções, apenas para conseguir lançar uma empresa concorrente à *OpenAI* durante essa pausa — algo que realmente aconteceu poucos meses depois, com o anúncio de sua nova empresa *xAI* em julho de 2023.

No entanto, a carta também foi assinada por cientistas notavelmente renomados das áreas de computação e IA, como Yoshua Bengio (professor da Universidade de Montreal e, assim como Yann LeCun e Geoffrey Hinton, vencedor do Prêmio Turing e considerado um dos pais da área de Inteligência Artificial e *Deep Learning*), Stuart Russell (professor de computação na Universidade de Berkeley e coautor do famoso livro *Inteligência Artificial: Uma Abordagem Moderna*), Emad Mostaque (fundador e CEO da *Stability AI*, empresa por trás do modelo generativo de imagens *Stable Diffusion*), Steve Wozniak (cofundador da *Apple*), John J. Hopfield (professor emérito da Universidade de Princeton e inventor das redes neurais Hopfield), entre outros.

Olhando em retrospecto a partir de 2025, o episódio se revela quase irônico: não só a pausa nunca aconteceu, como a corrida pela supremacia em IA se acelerou drasticamente nos meses seguintes.

Contudo, vale ressaltar que a carta aberta ajudou a iniciar e propagar a conversa sobre os riscos da IA. Dois meses após a sua publicação, o *Center for AI Safety* (CAIS), uma organização sem fins lucrativos, soltou uma declaração ainda mais forte sobre os possíveis riscos da Inteligência Artificial, com a finalidade de seguir com essa conversa:

> *A mitigação do risco de extinção provocado pela IA deve ser uma prioridade global, juntamente com outros riscos de escala social, como pandemias e a guerra nuclear.* [39]

Essa declaração foi assinada por alguns dos cientistas que haviam assinado a carta aberta, como Yoshua Bengio e Stuart Russell, mas também por outras figuras extremamente relevantes nas área de tecnologia e Inteligência Artificial, como o já mencionado Geoffrey Hinton, Demis Hassabis (CEO do *Google DeepMind*), Sam Altman (CEO da *OpenAI*), Dario Amodeo (CEO da *Anthropic*), Bill Gates (fundador da *Microsoft*), Ilya Sutskever (cofundador e cientista-chefe da *OpenAI*), Mira Murati (CTO da *OpenAI*), Igor Babuschkin (cofundador da *xAI*), Kevin Scott (CTO da *Microsoft*) e muitos outros.

A tensão entre o avanço acelerado e a segurança ficou evidente dentro da própria *OpenAI*. Em meados de 2023, a empresa anunciou com grande alarde a criação de uma equipe de "*Superalinhamento*", prometendo dedicar 20% de seu poder computacional para garantir o controle de futuras IAs superinteligentes. No entanto, menos de um ano depois, em maio de 2024, seus líderes (o cofundador e cientista-chefe Ilya Sutskever e o pesquisador de alinhamento Jan Leike) deixaram a empresa.

Jan Leike declarou publicamente que havia chegado a um "ponto de ruptura", afirmando que a cultura interna passou a priorizar "produtos brilhantes" em detrimento dos processos de segurança. Esse episódio dramático expôs as crises reais dentro dos principais

laboratórios de IA e a dificuldade de equilibrar a intensa corrida comercial das empresas com a precaução sobre o futuro.

Ou não seria o fim do mundo?

Tudo isso pode levar muitas pessoas a entrarem em pânico ou, no mínimo, a acenderem uma luz de alerta. Porém, há também os especialistas que são mais otimistas e não concordam com essa visão apocalíptica sobre o futuro das IAs.

Dentre os três nomes mencionados como "vencedor do Prêmio Turing" e "pai da área de Inteligência Artificial e *Deep Learning*", um deles, Yann LeCun, atualmente cientista-chefe de IA na *Meta*, acredita que o mero conceito de que IAs representariam uma ameaça grave à humanidade seria uma ideia absurdamente ridícula. Além disso, diz que, apesar de ter certeza de que as IAs vão ultrapassar a inteligência humana, que podemos confiar nos cientistas e especialistas responsáveis por criá-las de maneira segura; e que, se um dia percebermos que elas não são seguras, elas estarão rodando em algum *data center*, com um botão de "desligar"[40] .

Seguindo esse otimismo, em outubro de 2023, surgiu uma luz na escuridão que era o comportamento dos modelos de redes neurais profundas e *LLMs*. Conforme mencionado anteriormente neste e em outros capítulos, a matemática por trás das redes neurais é amplamente conhecida, pois cada neurônio realiza operações aritméticas relativamente simples. Porém, a interpretação de como esses neurônios se relacionam e o motivo pelo qual essas operações matemáticas chegam aos resultados ainda é desconhecido — a já citada "caixa preta".

Pois bem, na data citada acima, a empresa *Anthropic*, desenvolvedora do *chatbot Claude*, um dos concorrentes diretos do *ChatGPT*, lançou um artigo científico chamado "*Towards Monosemanticity: Decomposing Language Models With Dictionary Learning*" (traduzindo, "Rumo à Monossemanticidade: Decompondo

Modelos de Linguagem com Aprendizado de Dicionário")[41] com uma importante descoberta.

Assim como no caso do cérebro humano, em uma rede neural, neurônios individuais não têm relações consistentes com o comportamento da rede. Por exemplo, um único neurônio pode ser ativado em diversos contextos não relacionados. Por esse motivo, os pesquisadores sugeriram que, em vez de focar nos neurônios individuais, seria melhor analisar outros tipos de unidades, e eles conseguiram identificar um tipo de unidade que chamaram de "*features*" (que não tem uma tradução exata, mas podemos traduzir de maneira aproximada como "atributos" ou "propriedades"), que correspondem a padrões (combinações lineares) de ativação dos neurônios.

Em um modelo de linguagem relativamente pequeno baseado na arquitetura de *Transformers*, eles conseguiram decompor uma camada com 512 neurônios em mais de 4000 dessas *features*, que, separadamente, representavam coisas como sequências de DNA, linguagem jurídica, textos em hebraico, entre outros.

Para confirmar que essas *features* eram mais facilmente interpretáveis do que os neurônios, os pesquisadores realizaram uma avaliação com seres humanos e também utilizaram uma abordagem de "autointerpretabilidade", na qual um *LLM* tinha que gerar descrições curtas das *features* desse modelo. Ambos os métodos confirmaram que as *features* eram, de fato, mais interpretáveis do que os neurônios individuais.

Além disso, os pesquisadores notaram que, ao ativar uma *feature* de maneira artificial, o comportamento do modelo mudava exatamente da maneira esperada, e que, analisando o conjunto de *features* de maneira mais geral e abrangente, as *features* aprendidas eram majoritariamente universais entre modelos diferentes — ou seja, o que conseguirmos aprender ao estudar as *features* de um modelo provavelmente poderá ser generalizado para outros modelos.

Claro, esse artigo foi feito em cima de um modelo de linguagem de tamanho reduzido, então ainda há um longo caminho pela frente até chegarmos a resultados parecidos para *LLMs* colossais como o *GPT-4*, o *Claude*, o *Gemini* e o *LLaMa*. De qualquer forma, esse é um primeiro passo importantíssimo na busca para alcançarmos modelos de IA mais inteligentes, transparentes, confiáveis e seguros para todos nós.

Termino este tópico parafraseando a própria equipe da *Anthropic* com uma sentença que serve também como uma promessa de otimismo para o futuro do desenvolvimento de modelos de IA: pela primeira vez, é possível sentir que o principal desafio para podermos interpretar grandes modelos de linguagem é uma questão de engenharia, e não de ciência.

9.7 Água, energia e o custo ambiental

Enquanto as discussões sobre IA frequentemente habitam o mundo abstrato dos softwares, algoritmos e da "nuvem", existe um segredo muito mais terreno: existe um custo físico massivo por trás dessa revolução. Por trás de cada resposta do *ChatGPT* e de cada imagem gerada pelo *Midjourney*, há uma infraestrutura colossal de *data centers* que consomem quantidades significativas de dois dos nossos recursos mais preciosos: água e energia.

Água

Um dos custos físicos mais diretos da IA é o seu consumo de água. Modelos de linguagem exigem um poder computacional imenso, e os processadores que rodam esses sistemas geram uma quantidade de calor absurda. Para evitar que esses servidores superaqueçam e parem de funcionar, eles precisam ser constantemente resfriados, um processo que, na maioria das vezes, utiliza enormes volumes de água.

Para quantificar essa demanda, um estudo de 2023 da Universidade da Califórnia Riverside estimou que treinar o modelo *GPT-3* da *OpenAI* consumiu cerca de 700.000 litros de água[42] . No uso diário, os pesquisadores calcularam que uma conversa com 20 a 50 prompts no ChatGPT pode consumir o equivalente a uma garrafa de 500 ml de água. Cada resposta obtida de um modelo gasta, em média, 15 ml de água.

No entanto, é crucial colocar esses números em perspectiva. Embora os bilhões de litros consumidos pela indústria de tecnologia representem um volume enorme e crescente, eles ainda são uma fração do consumo de outros setores. A agricultura, por exemplo, é de longe o maior consumidor de água doce do planeta. Segundo o Banco Mundial, o setor agrícola é responsável por aproximadamente 70% de toda a retirada de água doce de rios, lagos e aquíferos globalmente[43] . Apenas a produção de um quilo de carne bovina pode consumir mais de 15.000 litros de água, e um quilo de abacate, cerca de 2.000 litros. Comparado a isso, o consumo direto da indústria de IA, embora significativo, opera em uma escala muito menor.

Então, onde reside a verdadeira preocupação? O problema (ainda) não é tanto o volume absoluto em escala global, mas sim o estresse hídrico localizado. Um *data center* não consome água de um reservatório global abstrato, ele a retira de bacias hidrográficas locais, competindo diretamente com as necessidades da agricultura e das populações daquela região. A construção de novos *data centers* em áreas que já enfrentam escassez de água, como o Arizona nos Estados Unidos (um local popular para esses centros), tem um impacto desproporcionalmente alto na comunidade local, mesmo que seu consumo seja uma "gota no oceano" a nível mundial. A OCDE, por exemplo, estima que o consumo de água pela indústria de IA poderá atingir até 6,6 bilhões de metros cúbicos por ano até 2027, o que seria o dobro do consumo anual de um país como a Suíça[44] . Esses dados reforçam que, embora não seja o maior consumidor em escala global, o crescimento exponencial da IA

a torna um vetor de risco significativo para a segurança hídrica em nível regional e um ponto crítico para a sustentabilidade da própria indústria.

A questão ética, portanto, não é se a IA vai "secar o planeta", mas sim uma questão de responsabilidade social e geográfica: como garantir que o avanço tecnológico não aconteça às custas da segurança hídrica das comunidades que hospedam sua infraestrutura?

Energia

Se a sede da IA é uma questão de impacto localizado, sua fome por energia representa um desafio de escala global. A Agência Internacional de Energia (IEA) projetou, em seu relatório de 2024, que o consumo elétrico dos *data centers*, impulsionado pela IA, tem o potencial de dobrar até 2026[45] . A demanda adicional seria equivalente a todo o consumo de energia do Japão.

Novamente, o contexto é fundamental. O consumo global de energia dos *data centers* ainda é uma pequena fração (cerca de 1-2%) do total mundial. No entanto, o problema não está no número absoluto de hoje, mas na velocidade do seu crescimento. Em um mundo que luta para fazer a transição para fontes de energia limpa, a súbita aparição de um consumidor tão voraz pode pressionar as redes elétricas e, no curto prazo, aumentar a dependência de combustíveis fósseis para suprir os picos de demanda. Grandes empresas de tecnologia, como a *Microsoft* e o *Google*, já admitem que a expansão da IA está tornando mais difícil cumprir suas próprias metas de neutralidade de carbono.

É nesse cenário que as gigantes da tecnologia começaram a fazer movimentos estratégicos ousados para garantir uma fonte de energia que seja, ao mesmo tempo, massiva, constante e com baixa emissão de carbono. A solução encontrada por algumas delas aponta para uma das tecnologias mais potentes, limpas e controversas da nossa era: a energia nuclear.

Em junho de 2025, a *Amazon* anunciou que investirá 20 bilhões de dólares na construção de dois *data centers* na Pensilvânia, nos Estados Unidos[46] . Além disso, a empresa fez um acordo com a *Talen Energy* até 2042, companhia responsável pela usina nuclear de *Susquehanna*, que alimentará um dos novos *data centers*[47] . A *Amazon* não comprou a usina, mas o acordo garante que o *data center* terá acesso direto a uma fonte de energia constante, praticamente comprando a produção inteira de um dos reatores.

Outras empresas olham para uma fronteira ainda mais futurista. A *Microsoft* assinou um acordo com a startup de fusão nuclear *Helion*, na qual Sam Altman, CEO da *OpenAI*, é um dos maiores investidores. O objetivo é que a *Helion* forneça eletricidade gerada por fusão nuclear para a *Microsoft* a partir de 2028[48] . A fusão nuclear, o processo que alimenta o Sol, é vista por muitos como o "santo graal" da energia limpa, mas ainda é uma tecnologia experimental. O fato de a *Microsoft* estar apostando nela demonstra o tamanho do desafio energético que a IA representa.

A questão ética que se impõe é mais sobre os *trade-offs* que estamos dispostos a aceitar: para desenvolver uma Inteligência Artificial cada vez mais poderosa, estamos abrindo as portas para uma nova era de energia nuclear em larga escala, com todas as responsabilidades e riscos que ela acarreta?

A decisão de investir em energia nuclear, seja na fissão (a tecnologia atual) ou na fusão (a promessa futura), não é uma solução sem desafios. As complexidades envolvendo segurança, gestão de resíduos e aceitação pública continuam sendo barreiras significativas. Porém, a disposição da indústria de IA em investir bilhões nessa área demonstra o quão sério é o desafio energético nessa área.

Com isso, o avanço da Inteligência Artificial se torna intrinsecamente ligado ao futuro da energia nuclear, potencialmente criando uma

codependência entre duas das tecnologias mais transformadoras e controversas dos nossos tempos.

9.8 Mercado de trabalho

A popularização dos modelos de IA generativa causaram um furor na mídia e no mercado de trabalho. Imediatamente, foi visto um grande *boom* de investimentos em praticamente qualquer tipo de empresa que utilizasse as palavras-chaves "Inteligência Artificial" (uma bolha, quiçá?). Porém, como discutimos no início do capítulo, tecnologias novas sempre geram medo ou apreensão a certos grupos de pessoas vivendo durante aquela geração, e um dos tópicos mais abordados foi a da possível extinção de diversas profissões e a substituição de humanos por máquinas.

Seria fácil reduzir essa discussão sobre a IA e o mercado de trabalho a uma pura matemática de subtração: IA menos empregos de humanos, resultando em desemprego em massa e descontentamento. Porém, vamos tentar não cair em lugares-comuns e explorar o assunto mais a fundo.

Não podemos dizer que esse medo dos trabalhadores é infundado. A modernização dos elevadores realmente acabou com a profissão de ascensorista. A popularização dos computadores pessoais realmente acabou com a profissão de datilógrafo. Mais recentemente, isso tem sido visto com a paulatina adoção de robôs em galpões e na indústria. Um estudo de 2020 de professores do *MIT* e da Universidade de Boston demonstrou que, nos Estados Unidos, para cada robô adicionado para um grupo de 1000 funcionários, os salários dos funcionários diminuíram em 0,42% e a taxa de empregabilidade da população caiu 0,2%, o que equivale a cerca de 400.000 empregos[49] .

Um caso que é constantemente citado em ambos os lados da discussão é o da *Amazon*. Em 2012, a empresa adquiriu uma empresa de desenvolvimento de robôs para armazéns chamada *Kiva Systems* e, desde então, vem ampliando sua quantidade e uso, possuindo mais de 500.000 funcionários-robôs atualmente. O sentimento geral era de que essa atitude eliminaria completamente a necessidade de funcionários humanos, mas a empresa se defende dizendo que, desde então, criou mais de 1 milhão de novas vagas para humanos, e que o uso de robôs apenas aumenta a produtividade dos galpões[50] . Ao mesmo tempo, há quem diga que essa coexistência é apenas temporária, e que até 2030 o número de robôs poderia passar o de seres humanos, com grande número de demissões.

E como isso se aplicará ao mundo dos *LLMs* e das IAs generativas?

Um estudo foi feito pela *OpenAI* e pela Universidade da Pennsylvania na época do lançamento do *GPT-4* tentando responder à pergunta: "qual é o possível impacto dos *LLMs* no mercado de trabalho?"[51] , com foco no mercado estadunidense. Eles chegaram à estimativa de que 80% dos trabalhadores vão ter pelo menos 10% de suas tarefas afetadas por modelos generativos, e que 19% dos trabalhadores podem ter até metade de suas tarefas impactadas por esses modelos.

Algumas das profissões listadas como potencialmente mais afetadas seriam: matemáticos, analistas financeiros e de impostos, escritores, tradutores, contadores, e até mesmo profissões de salários mais altos, como programadores, analistas de dados e engenheiros de *blockchain*. Quanto às habilidades, empregos que dependam da escrita e da programação são vistos como mais impactados, enquanto aqueles mais relacionados à ciência e ao pensamento críticos foram vistos como mais seguros. Além disso, trabalhos mais manuais como cozinheiros, encanadores e mecânicos também são vistos como mais seguros no que diz respeito à influência dos *LLMs*.

Temos também os relatórios publicados frequentemente pela *Anthropic*[52] , trazendo uma perspectiva ainda mais detalhada sobre esse impacto no mercado de trabalho. Em vez de focar apenas na substituição de empregos, a pesquisa introduziu o conceito de "exposição à IA", medindo o grau em que as tarefas de uma profissão podem ser executadas ou assistidas por modelos como o *Claude*. Os resultados confirmam que, ao contrário das ondas de automação anteriores, que atingiram principalmente o trabalho manual, as IAs generativas afetam desproporcionalmente as profissões de mais alta renda e que exigem maior formação educacional, como analistas financeiros, programadores, pesquisadores e escritores.

No entanto, a principal conclusão da *Anthropic* não é a substituição em massa, mas sim uma profunda redefinição do fluxo de trabalho. Para muitos desses profissionais, a IA não age como um substituto, mas como um "colega de trabalho" ou um "assistente capacitado", automatizando tarefas repetitivas (como a elaboração de rascunhos ou a análise de grandes volumes de dados) e liberando os humanos para se concentrarem em estratégia, revisão crítica e criatividade. Isso evidencia que a principal mudança no mercado de trabalho não é apenas o desaparecimento de cargos, mas a crescente demanda por novas habilidades de colaboração humano-IA, como a curadoria e verificação de resultados e a Engenharia de *Prompt* avançada.

Outras profissões que correm muito risco são a de *designer* e de artista gráfico. Além da questão de direitos autorais já mencionada neste capítulo, uma grande preocupação desses artistas é questão da precarização do trabalho. Com modelos de geração de imagem da capacidade do *Midjourney*, *FLEX* e várias ferramentas da *Adobe*, que geram imagens ultrarrealistas em segundos, eles temem se tornar irrelevantes e questionam por que as pessoas pagariam pelas suas artes quando elas próprias podem gerar algo de grande qualidade por um preço muito mais acessível e em um tempo muito mais curto.

Uma notícia de abril de 2023 diz que a quantidade de vagas para *designers* em empresas chinesas de jogos caiu em 70% desde o ano anterior, e um dos artistas abordados afirmou que, anteriormente, precisava de um dia inteiro para desenhar uma cena ou personagem; agora, com a ajuda desses modelos generativos, pode criar até 40 por dia[53] .

Talvez o exemplo mais concreto e ruidoso desse embate tenha sido as greves dos roteiristas (WGA) e atores (SAG-AFTRA) de Hollywood em 2023. Pela primeira vez, uma indústria inteira paralisou suas atividades, tendo a IA como uma de suas pautas centrais. Os profissionais exigiram (e conquistaram) cláusulas contratuais que regulam o uso de IAs generativas na criação de roteiros e estabelecem regras sobre o consentimento e a compensação para a criação e uso de "réplicas digitais" de atores. As greves de Hollywood se tornaram um marco, demonstrando que a negociação coletiva pode ser uma ferramenta poderosa para que os trabalhadores estabeleçam limites e garantam proteções.

Yann LeCun declarou que, apesar dessa possibilidade de empregos serem substituídos devido às IAs, as pessoas se tornam mais produtivas e o total de riqueza produzida por hora aumenta no longo prazo, mas que muitas profissões desaparecerão, novas serão criadas e essa adaptação pode demorar algo como 15 a 20 anos para ser vista na sociedade[54] .

Esse enorme aumento de produtividade é também uma questão controversa. Muitas empresas, como a própria *Amazon*, afirmam que, como os robôs e IAs estarão se ocupando dos trabalhos mais repetitivos, os humanos terão mais tempo para serem mais produtivos em trabalhos mais interessantes e desafiadores. Entretanto, há um outro lado a ser considerado.

Guilherme Silveira, cofundador e *Chief Education Officer* da Alura, empresa líder em educação tecnológica no Brasil e uma das primeiras a implementar IAs generativas em diversos processos

educacionais (como a *Luri*, um *chatbot* de IA para auxiliar no estudo dos alunos), exemplificou:

"*Desde que a computação de forma digital foi inventada, ela cria ondas de novas ferramentas para a humanidade. Agora não é diferente. Quando temos uma ferramenta nova, é fácil e imediato pensarmos: 'Como eu posso aumentar a produtividade de um ser humano através dessa ferramenta?', e isso, em geral, está muito ligado à questão de pensar no ser humano como uma máquina de produção. Ele está produzindo mais e mais.*

No mercado de trabalho, isso é uma forma de pensar muito imediata na nossa cabeça, pois é muito comum que estejamos pensando apenas nos resultados do trabalho, e a medida mais comum desse resultado é através de 'quantidades produzidas'. Muitas das ferramentas que surgem em cima de IAs generativas focam em automatizar processos repetitivos, sejam eles maquinais, computacionais e até mesmo criativos.

Porém, o ser humano não se resume à sua produção em um mercado de trabalho. Existem diversas outras características e capacidades humanas com as quais essas ferramentas podem nos ajudar. Veja como psicólogos e psicólogas utilizam diversas ferramentas e até mesmo jogos de tabuleiro, por exemplo, para focar no desenvolvimento humano. Não é a questão de o paciente ser uma máquina que deve produzir mais.

No mundo dos relacionamentos humanos, poderíamos pensar no lado da produtividade: usar uma ferramenta de IA para que o meu aplicativo maximize a chance de que um usuário encontre uma pessoa que seja parecida com ela, ou então, sugerir quais fotos maximizam a chance de obter um 'match' com uma possível pessoa parceira. IAs são utilizadas dessa forma já tem muitos anos. Essa maximização é produção, é um pensamento mercadológico imediato.

Como poderíamos pensar no lado humano em aplicativos de relacionamento? Seria pensar como a ferramenta pode fazer uma pessoa se tornar mais hábil em manter um relacionamento a

longo prazo, por exemplo. Um aplicativo que fizesse isso se distanciaria da questão de 'aumentar o seu número de matches' (desenvolvendo a pessoa apenas como uma 'máquina de matches') e se aproximaria da questão de 'desenvolver' as pessoas como seres humanos."

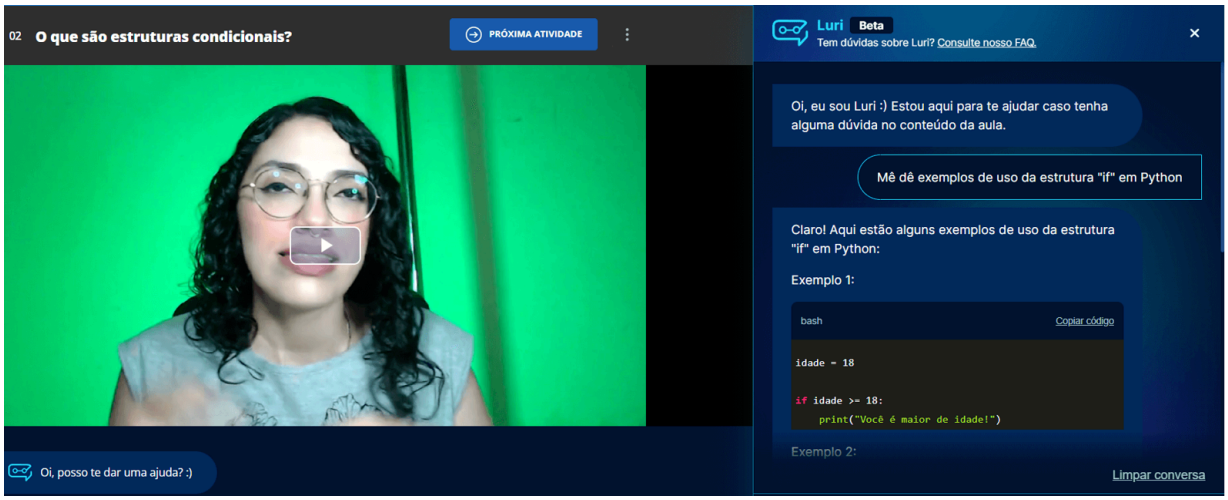

Figura 9.7: Exemplo de uso da Luri, a IA presente como assistente dentro dos cursos da Alura

Isso se conecta com outro ponto de atenção, que é a desumanização do trabalho. Na medicina, por exemplo, quem não gostaria de ter uma IA que conseguisse analisar seus exames e dar uma resposta precisa instantaneamente? Ao mesmo tempo, uma boa parte do tratamento, especialmente para doenças mais graves, passa pela relação pessoal do médico com o paciente.

Por esse e outros motivos, algo que é repetido exaustivamente nesse meio é que as ferramentas de IA são exatamente isso, **ferramentas**, e que o futuro do trabalho envolverá uma relação simbiótica entre humanos e máquinas, onde os humanos serão habilitados para trabalhar ao lado de IAs, extraindo o melhor de suas capacidades computacionais e analíticas e combinando-as com habilidades humanas como empatia, ética e pensamento crítico.

Scott Galloway, professor da Universidade de Nova Iorque, tem uma citação também frequentemente repetida. Parafraseando:

> "*A IA não vai roubar o seu emprego, mas alguém que saiba usar IAs vai.*"

Essa afirmação conversa com o ponto trazido por Yann LeCun e mencionado no início do tópico: novas tecnologias podem, sim, acabar com determinadas profissões, causar uma disrupção no mercado de trabalho e até mesmo crises financeiras, e todos nós, como sociedade, teremos que passar por um período de adaptação a essas inovações. Nesse quesito, os governos do mundo poderão (ou deverão) trabalhar junto das empresas de IA nas questões de regulamentação, como as pedidas por diversas companhias e já citadas anteriormente, e ditar as diretrizes para uma transição mais suave.

A adoção de IA deverá ser feita de forma que mitigue os possíveis problemas do desemprego tecnológico e o alargamento da divisão digital, e será necessário planejar estratégias eficazes de requalificação e reciclagem, garantindo que ninguém seja deixado para trás na marcha em direção a essa nova era.

Muitos meses antes do lançamento do *ChatGPT*, em 2021, Sam Altman escreveu um ensaio no qual propôs que, em 10 anos, a Inteligência Artificial poderia gerar riqueza o suficiente para pagar 13.500 dólares por ano a cada cidadão estadunidense[55] . Isso se refere ao conceito de "renda universal básica", outra questão que voltou à tona com muita força com a popularização das IAs, segundo o qual cada pessoa receberia um valor de dividendos da riqueza acumulada dessas empresas, a fim de diminuir a desigualdade social.

Essa questão da renda universal básica também gera muita polêmica, principalmente de diferentes lados políticos, mas é consenso que, de uma maneira ou de outra, teremos que nos

adaptar a essa nova realidade, promover o aprendizado inclusivo e contínuo e não deixar o lado humano de fora dessa nova equação.

Por mais sofisticada que seja a IA, ela é uma ferramenta, e como todas as ferramentas, deve ser usada com cuidado, consideração e sem alarmismos apocalípticos, pois, no papel de pessoas que vão desenvolvedor, operar e se beneficiar desses sistemas, a responsabilidade sobre eles recairá sobre todos nós.

Referências bibliográficas:

[1] DAVIS, Wes; LAWLER, Richard. Elon Musk blames data scraping by AI startups for his new paywalls on reading tweets. *The Verge*, 2023. Disponível em: https://www.theverge.com/2023/7/1/23781198/twitter-daily-reading-limit-elon-musk-verified-paywall

[2] SHAKIR, Umar. Reddit's upcoming API changes will make AI companies pony up. *The Verge*, 2023. Disponível em: https://www.theverge.com/2023/4/18/23688463/reddit-developer-api-terms-change-monetization-ai

[3] WEATHERBED, Jess. Google confirms it's training Bard on scraped web data, too. *The Verge*, 2023. Disponível em: https://www.theverge.com/2023/7/5/23784257/google-ai-bard-privacy-policy-train-web-scraping

[4] DAVID, Emilia. Now you can block OpenAI's web

crawler. *The Verge*, 2023. Disponível em: https://www.theverge.com/2023/8/7/23823046/openai-data-scrape-block-ai

[5] ROSSOW, Andrew. X Updates its Terms, Bans Data Scraping& Crawling. *NFT Now*, 2023. Disponível em: https://nftnow.com/news/x-updates-terms-of-service-to-ban-unauthorized-data-crawling-scraping/

[6] RAY, Siladitya. Apple Joins A Growing List Of Companies Cracking Down On Use Of ChatGPT By Staffers—Here's Why. *Forbes*, 2023. Disponível em: https://www.forbes.com/sites/siladityaray/2023/05/19/apple-joins-a-growing-list-of-companies-cracking-down-on-use-of-chatgpt-by-staffers-heres-why/

[7] GURMAN, Mark. Samsung Bans Staff's AI Use After Spotting ChatGPT Data Leak. *Bloomberg*, 2023. Disponível em: https://www.bloomberg.com/news/articles/2023-05-02/samsung-bans-chatgpt-and-other-generative-ai-use-by-staff-after-leak/

[8] LIU, Yi; DENG, Gelei; XU, Zhengzi; LI, Yuekang; ZHENG, Yaowen; ZHANG, Ying; ZHAO, Lida; ZHANG, Tianwei; LIU, Yang. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *arXiv*, 2023.

Disponível em: https://arxiv.org/abs/2305.13860

[9] ZORTHIAN, Julia. OpenAI CEO Sam Altman Asks Congress to Regulate AI. *Time*, 2023. Disponível em: https://time.com/6280372/sam-altman-chatgpt-regulate-ai/

[10] PERRIGO, Billy. OpenAI Could Quit Europe Over New AI Rules, CEO Sam Altman Warns. *Time*, 2023. Disponível em: https://time.com/6282325/sam-altman-openai-eu/

[11] US tightens its grip on AI chip flows across the globe. *Reuters*, 2025. Disponível em: https://www.reuters.com/technology/artificial-intelligence/us-tightens-its-grip-ai-chip-flows-across-globe-2025-01-13/

[12] POTKIN, F. Huawei readies new AI chip for mass shipment as China seeks Nvidia alternatives, sources say. *Reuters*, 2025. Disponível em: https://www.reuters.com/world/china/huawei-readies-new-ai-chip-mass-shipment-china-seeks-nvidia-alternatives-sources-2025-04-21/

[13] Today we're open-sourcing R1 1776. *Perplexity AI*, 2025. Disponível em:

https://www.perplexity.ai/hub/blog/open-sourcing-r1-1776

[14] BAIO, Andy. Exploring 12 Million of the 2.3 Billion Images Used to Train Stable Diffusion's Image Generator. *Waxy.org*, 2022. Disponível em: https://waxy.org/2022/08/exploring-12-million-of-the-images-used-to-train-stable-diffusions-image-generator/

[15] ESCALANTE-DE MATTEI, Shanti. Artists Are Suing Artificial Intelligence Companies and the Lawsuit Could Upend Legal Precedents Around Art. *Art in America*, 2023. Disponível em: https://www.artnews.com/art-in-america/features/midjourney-ai-art-image-generators-lawsuit-1234665579/

[16] BRITTAIN, Blake. Getty Images lawsuit says Stability AI misused photos to train AI. *Reuters*, 2023. Disponível em: https://www.reuters.com/legal/getty-images-lawsuit-says-stability-ai-misused-photos-train-ai-2023-02-06/

[17] HURST, Luke. Rapid growth of 'news' sites using AI tools like ChatGPT is driving the spread of misinformation. *EuroNews*, 2023. Disponível em:

https://www.euronews.com/next/2023/05/02/rapid-growth-of-news-sites-using-ai-tools-like-chatgpt-is-driving-the-spread-of-misinforma

[18] SOARES, Lucas. ChatGPT é usado por bandidos para golpes sem erros de digitação. *Olhar Digital*, 2023. Disponível em: https://olhardigital.com.br/2023/04/02/seguranca/chatgpt-e-usado-por-bandidos-para-golpes-sem-erros-de-digitacao/

[19] FERREIRA, Yuri. Foto de papa de bobojaco estiloso foi feita com IA; Vogue Brasil caiu na fake news. *Fórum*, 2023. Disponível em: https://revistaforum.com.br/ciencia-e-tecnologia/2023/3/25/foto-de-papa-de-bobojaco-estiloso-foi-feita-com-ia-vogue-brasil-caiu-na-fake-news-133327.html

[20] COLE, Samantha. AI-Assisted Fake Porn Is Here and We're All Fucked. *Vice*, 2017. Disponível em: https://www.vice.com/en/article/gydydm/gal-gadot-fake-ai-porn

[21] ALTMAN, Sam. (@sama). "I am nervous about the impact AI is going to have on future elections (at least until everyone gets used to it). personalized 1:1

persuasion, combined with high-quality generated media, is going to be a powerful force." 3 ago. 2023, 19h52. Tweet. Disponível em: https://twitter.com/sama/status/1687236201496064000

[22] KELLY, Makena. Meta, Google, and OpenAI promise the White House they'll develop AI responsibly. *The Verge*, 2023. Disponível em: https://www.theverge.com/2023/7/21/23802274/artificial-intelligence-meta-google-openai-white-house-security-safety

[23] SynthID - A tool to watermark and identify content generated through AI. *Google*. Disponível em: http://deepmind.google/science/synthid/

[24] VIÉS. In: *WIKIPÉDIA*: a enciclopédia livre. Disponível em: https://pt.wikipedia.org/wiki/Vi%C3%A9s

[25] RODRIGUES, Léo. Relação com crédito bancário varia conforme classe social, diz estudo. *Agência Brasil*, 2023. Disponível em: https://agenciabrasil.ebc.com.br/economia/noticia/2023-08/relacao-com-credito-bancario-varia-conforme-classe-social-diz-estudo

[26] DASTIN, Jeffrey. Amazon scraps secret AI recruiting tool that showed bias against women. *Reuters*, 2018. Disponível em: https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G

[27] LARSON, Jeff; MATTU, Surya; KIRCHNER, Lauren; ANGWIN, Julia. How We Analyzed the COMPAS Recidivism Algorithm. *ProPublica*, 2016. Disponível em: https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm

[28] GRANT, Nico; HILL, Kashmir. Google's Photo App Still Can't Find Gorillas. And Neither Can Apple's. *The New York Times*, 2023. Disponível em: https://www.nytimes.com/2023/05/22/technology/ai-photo-labels-google-apple.html

[29] GRANT, N. Google Chatbot's A.I. Images Put People of Color in Nazi-Era Uniforms. *New York Times*, 2024. Disponível em: https://www.nytimes.com/2024/02/22/technology/google-gemini-german-uniforms.html

[30] Sycophancy in GPT-4o: what happened and what we're doing about it. *OpenAI*, 2025. Disponível em:

https://openai.com/index/sycophancy-in-gpt-4o/

[31] Elon Musk Claims Grok 'Manipulated' By X Users After Chatbot Praises Hitler. *Forbes*, 2025. Disponível em: https://www.forbes.com/sites/tylerroush/2025/07/09/elon-musk-claims-grok-manipulated-by-x-users-after-chatbot-praises-hitler/

[32] BOSTROM, Nick. Ethical Issues in Advanced Artificial Intelligence. *Cognitive, Emotive and Ethical Aspects of Decision Making in Humans and in Artificial Intelligence*, v. 2, ed. I, 2003. Disponível em: https://nickbostrom.com/ethics/ai

[33] MILES, Kathleen. Artificial Intelligence May Doom The Human Race Within A Century, Oxford Professor Says. *HuffPost*, 2014. Disponível em: https://www.huffpost.com/entry/artificial-intelligence-oxford_n_5689858

[34] BROWN, Sara. Why neural net pioneer Geoffrey Hinton is sounding the alarm on AI. *MIT Sloan*, 2023. Disponível em: https://mitsloan.mit.edu/ideas-made-to-matter/why-neural-net-pioneer-geoffrey-hinton-sounding-alarm-ai

[35] KNIGHT, Will. What Really Made Geoffrey Hinton Into an AI Doomer. *Wired*, 2023. Disponível em: https://www.wired.com/story/geoffrey-hinton-ai-chatgpt-dangers/

[36] BILLS, Steven; CAMARATA, Nick; MOSSING, Dan; TILLMAN, Henk; GAO, Leo; GOH, Gabriel; SUTSKEVER, Ilya; LEIKE, Jan; WU, Jeff; SAUNDERS, William. Language models can explain neurons in language models. *OpenAI*, 2023. Disponível em: https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html

[37] LEIKE, Jan; SCHULMAN, John; WU, Jeffrey. Our approach to alignment research. *OpenAI*, 2022. Disponível em: https://openai.com/blog/our-approach-to-alignment-research

[38] FUTURE of Life Institute. *Pause Giant AI Experiments*: An Open Letter. 2023. Disponível em: https://futureoflife.org/open-letter/pause-giant-ai-experiments/

[39] CENTER For AI Safety. *Statement on AI Risk*. 2023. Disponível em: https://www.safe.ai/statement-on-ai-risk

[40] TAYLOR, Chloe. Almost half of CEOs fear A.I. could destroy humanity 5 to 10 years from now—but one 'A.I. godfather' says an existential threat is 'preposterously ridiculous'. *Fortune*, 2023. Disponível em: https://fortune.com/2023/06/15/yann-lecun-ai-godfather-destroy-humanity-threat/

[41] BRICKEN, Trenton *et al*. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning. *Anthropic*, 2023. Disponível em: https://www.anthropic.com/index/towards-monosemanticity-decomposing-language-models-with-dictionary-learning

[42] LI, P. et al. Making AI Less "Thirsty": Uncovering and Addressing the Secret Water Footprint of AI Models. *arXiv*, 2025. Disponível em: https://arxiv.org/abs/2304.03271

[43] Water in Agriculture. *The World Bank*, 2024. Disponível em: https://documents1.worldbank.org/curated/en/875921614166983369/pdf/Water-in-Agriculture-Towards-Sustainable-Agriculture.pdf

[44] A Handbook of What Works - Solutions for the Local Implementation of the OECD Principles on

Water Governance. *OECD*, 2024. Disponível em: https://www.oecd.org/content/dam/oecd/en/publications/reports/2024/05/a-handbook-of-what-works_33112bf3/bf54627e-en.pdf

[45] Electricity 2024 - Analysis and forecast to 2026. *International Energy Agency (IEA)*, 2024. Disponível em: https://iea.blob.core.windows.net/assets/6b2fd954-2017-408e-bf08-952fdd62118a/Electricity2024-Analysisandforecastto2026.pdf

[46] LEVY, M. Amazon to spend $20B on data centers in Pennsylvania, including one next to a nuclear power plant. *AP News*, 2025. Disponível em: https://apnews.com/article/amazon-data-center-nuclear-power-plant-pennsylvania-electricity-grid-31f705d035069279b70fa27a5dc71596

[47] Talen Energy and Amazon sign nuclear power deal to fuel data centers. *Reuters*, 2025. Disponível em: https://www.reuters.com/business/energy/talen-energy-amazon-sign-nuclear-power-deal-fuel-data-centers-2025-06-11/

[48] Gardner, T. Microsoft signs power purchase deal with nuclear fusion company Helion. *Reuters*, 2023.

Disponível em: https://www.reuters.com/technology/microsoft-buy-power-nuclear-fusion-company-helion-2023-05-10/

[49] ACEMOGLU, Daron; RESTREPO, Pascual. Robots and Jobs: Evidence from U.S. Labor Markets. *Journal of Political Economy*, v. 128, n. 6, 2020. Disponível em: https://economics.mit.edu/sites/default/files/publications/Robots%20and%20Jobs%20-%20Evidence%20from%20US%20Labor%20Markets.p.pdf

[50] RUSHE, Dominic. Amazon enters the age of robots. What does that mean for its workers?. *The Guardian*, 2022. Disponível em: https://www.theguardian.com/technology/2022/nov/11/amazon-robots-jobs

[51] ELOUNDOU, Tyna; MANNING, Sam; MISHKIN, Pamela; ROCK, Daniel. GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models. *arXiv*, 2023. Disponível em: https://arxiv.org/abs/2303.10130

[52] Anthropic Economic Index - Understanding AI's effects on the economy over time. *Anthropic*.

Disponível em: https://www.anthropic.com/economic-index

[53] ZHOU, Viola. AI is already taking video game illustrators' jobs in China. *Rest of World*, 2023. Disponível em: https://restofworld.org/2023/ai-image-china-video-game-layoffs/

[54] GRALLET, Guillaume; PONS, Héloïse. Yuval Noah Harari (Sapiens) versus Yann Le Cun (Meta) on artificial intelligence. *Le Point*, 2023. Disponível em: https://www.lepoint.fr/sciences-nature/yuval-harari-sapiens-versus-yann-le-cun-meta-on-artificial-intelligence-11-05-2023-2519782_1924.php

[55] ALTMAN, Sam. Moore's Law for Everything. 2021. Disponível em: https://moores.samaltman.com/

Capítulo 10
Extra — Rodando modelos open source localmente com o LM Studio

Este é um capítulo extra apenas para pessoas entusiastas e que tenham um computador bom o suficiente.

Até agora, falamos de modelos de IA como se fossem entidades distantes, vivendo em supercomputadores caríssimos em algum lugar da Califórnia. Bom, não deixam de ser. Mas e se eu dissesse que você pode rodar um desses modelos de IA *open source* direto no seu computador? Sem depender de internet, sem pagar por *API* e, o mais importante, sem que ninguém fique de olho nas suas conversas.

A comunidade criou algumas ferramentas excelentes que permitem isso, dentre as quais as mais populares são o *Ollama* (https://ollama.com/) e o *LM Studio* (https://lmstudio.ai/) para computadores, e o *PocketPal* para dispositivos móveis (celulares e tablets).

Para este tutorial, vou seguir com o *LM Studio*, devido à sua interface visual mais amigável, e com o modelo *Gemma 3* do *Google*, que tem versões em diferentes tamanhos. É um software gratuito e fácil de usar, que permite baixar e usar modelos *open source* como o *Gemma*, o *Llama*, o *Phi*, o *SmolLM*, entre outros, de forma local, privada e offline.

10.1 Preparando o sistema

Antes de tudo, certifique-se de que seu computador atende aos requisitos mínimos. Um processador moderno é essencial. É recomendado ter pelo menos 16 GB de RAM, embora 32 GB ofereça mais estabilidade e velocidade, especialmente ao rodar modelos maiores. Uma *GPU* dedicada ajuda, mas não é obrigatória para modelos pequenos.

O primeiro passo é baixar o *LM Studio*. Acesse o site https://lmstudio.ai/ e escolha a versão compatível com seu sistema operacional. Daí, é só instalá-lo normalmente. A carinha dele é mais ou menos assim:

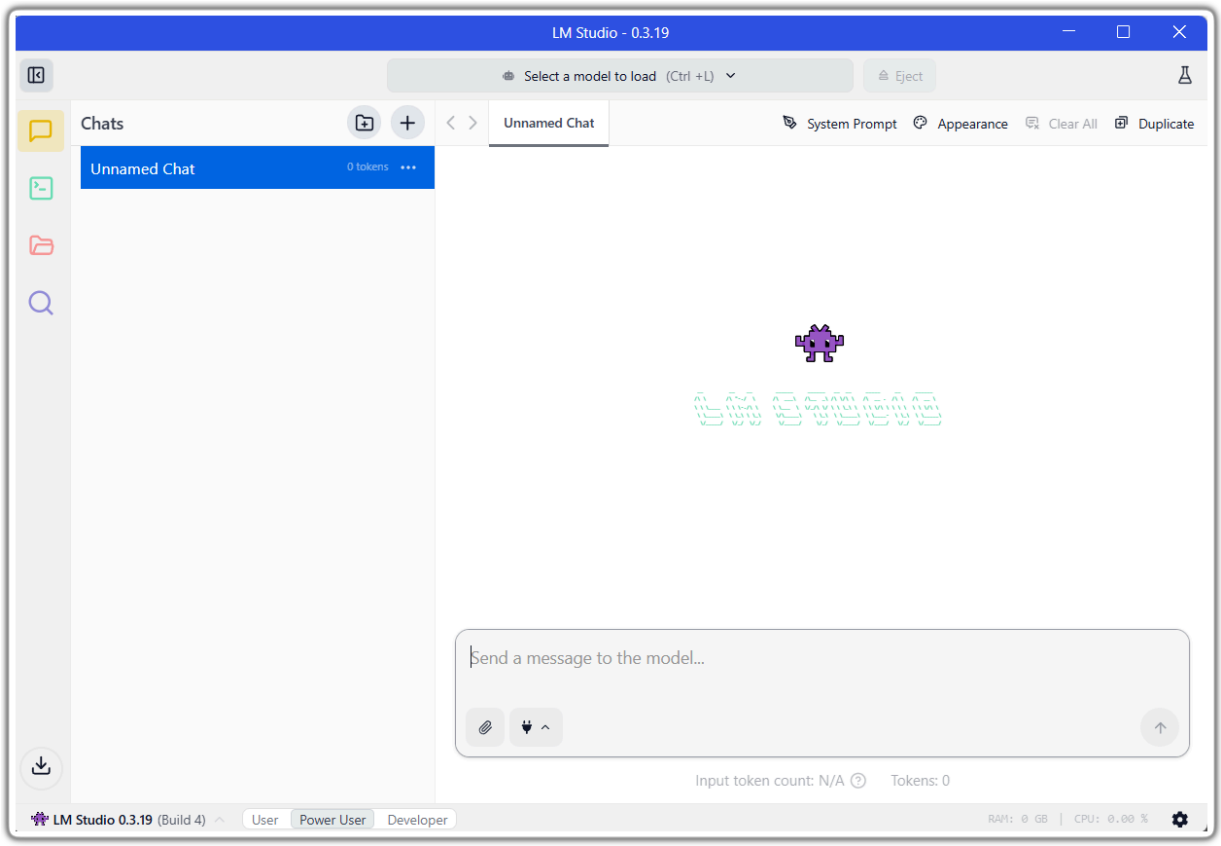

Figura 10.1: Tela inicial do LM Studio

10.2 Baixando o modelo

Uma das grandes facilidades dessa ferramenta é que você pode fazer o download dos modelos *open source* dentro dela própria.

Com o *LM Studio* aberto, vá até a aba "*Discover*" do lado esquerdo e digite "**Gemma 3**" no campo de busca. Você verá diversas versões do modelo, com diferentes quantizações, feitas tanto oficialmente pelas empresas criadoras desses modelos quanto pela comunidade de IA *open source*. O *Google* é uma das empresas que cria e disponibiliza as suas próprias quantizações (mas, caso queira testar modelos de outras empresas e não encontre uma quantização oficial, recomendo as criadas pela empresa *Unsloth* ou pela *lmstudio-community*).

A minha máquina tem uma boa performance, então vou escolher a versão "*Gemma 3 4B*", ou seja, é um modelo de 4 bilhões de parâmetros. Caso sua máquina não aguente rodá-lo, recomendo baixar um modelo menor, como o "*Gemma 3 1B*", o "*Gemma 2 2B*", o "*SmolLM3-3B*" ou o "*Qwen3 1.7B*".

Além disso, já dentro da página do modelo escolhido, em "*Download Options*", ao clicar em cima do nome do modelo você notará algumas sequências de letras ao lado dele, como "*Q4_K_M*", "*Q6_K*", "*Q8_0*", entre outras. Isso se refere à qualidade da quantização que foi aplicada a esse modelo (lembra da quantização que vimos no capítulo 8?), mais especificamente, ao número de bits. Números maiores representam um arquivo final maior, menos "enxugado", mais próximo do original; números menores, um arquivo final menor, mais "enxugado", que pode ter uma performance mais distante do original.

Pense nisso como assistir um vídeo no *YouTube* em 720p ou em 4K: sim, a imagem do 4K é maravilhosa, mas às vezes a sua internet não está nos melhores dias, e uma qualidade de 720p para ver um vídeo de gatos pulando ao verem pepinos é mais do que o suficiente.

Para um bom equilíbrio entre desempenho e qualidade, vamos escolher uma versão quantizada em 4 bits, a "**Q4_K_M**". Essa versão é leve o suficiente para rodar em muitos *PCs* comuns sem sacrificar tanto a capacidade do modelo.

> **DETALHE TÉCNICO:**
>
> Na prática, o "Q4_K_M" comprime o modelo para uma média de 4.5 bits por parâmetro. A grande maioria dos dados do modelo (os "pesos" numéricos) é de fato comprimida para 4 bits cada, mas para evitar uma grande perda de qualidade, esse formato agrupa os pesos em "blocos". Para cada bloco, ele armazena metadados adicionais de alta precisão (chamadas de "fatores de escala" e "mínimos"). Esses metadados funcionam como um "gabarito" que ajuda o modelo a descompactar os valores de 4 bits de forma muito mais precisa.

Clique em "*Download*" e espere. O *Gemma 3* de 4B tem em torno de 3.3GB, já o de 1B tem meros 720MB. Ao final do download, o *LM Studio* vai armazenar o modelo localmente, pronto para ser usado *offline*.

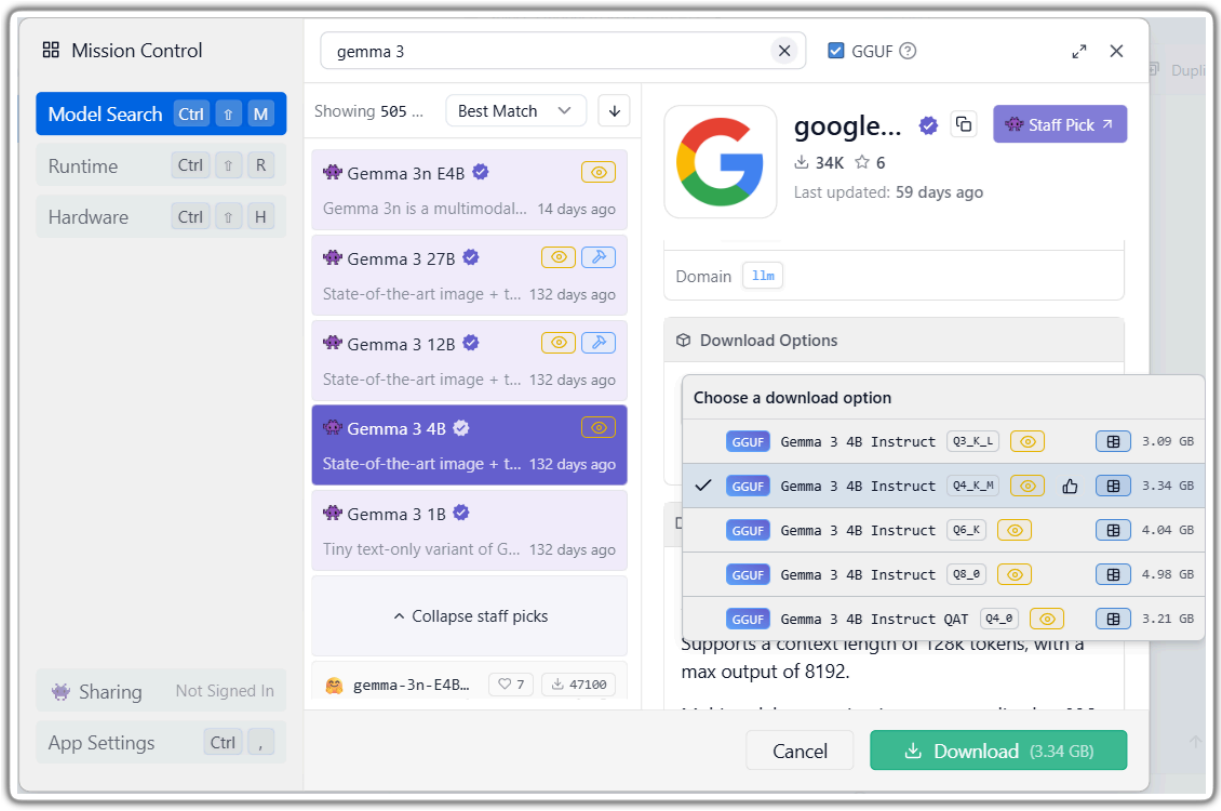

Figura 10.2: Tela de downloads do LM Studio com as diferentes quantizações do Gemma 3

10.3 Rodando o modelo

Download concluído? Ótimo. Agora, você tem duas opções. Você pode simplesmente clicar no botão "*Load Model*", que apareceu ao final do download:

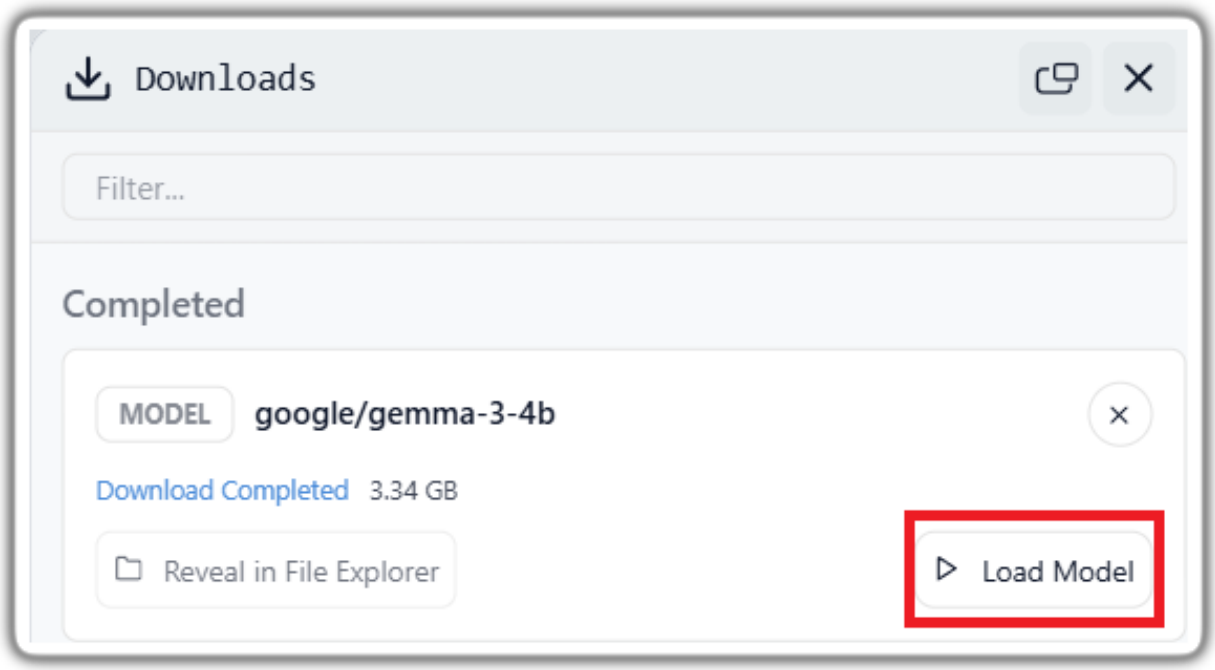

Figura 10.3: Download terminado no LM Studio, já é possível carregar o modelo no botão 'Load'

Ou ir para a aba de "Chat" (o ícone de balão de conversa à esquerda). No topo da tela, você verá um menu com o texto "*Select a model to load*" para selecionar o modelo:

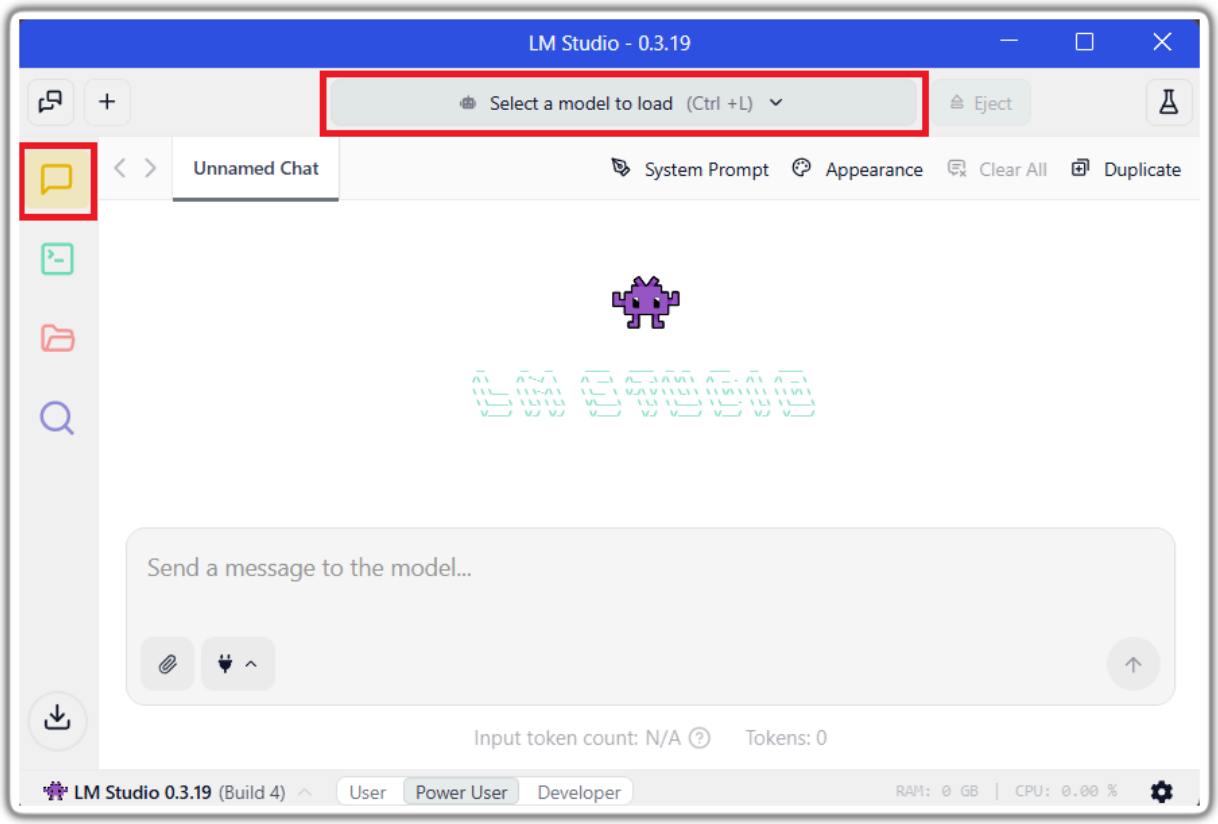

Figura 10.4: Tela de 'Chat' do LM Studio, destacando o campo de seleção de modelos

Escolha o "*Gemma 3*" que você acabou de baixar. O *LM Studio* vai carregá-lo na memória, o que pode levar alguns segundos (isso considerando que a sua máquina vai aguentar). Caso não aguente, ~~compre uma nova~~ tente com um modelo menor.

Você verá uma barrinha de progresso e, quando ela sumir... pronto! O palco é seu. A IA está carregada e esperando na janela de chat, que funcionará exatamente como o *ChatGPT*, o *Gemini*, o *Claude* ou qualquer *chatbot* com o qual você esteja acostumado. Como o "*Gemma 3 4B*" é um modelo multimodal, além de conversar por texto, você consegue até mandar imagens para ele analisar.

Mande um "oi", pergunte sobre o sentido da vida, peça uma receita de bolo de fubá. A conversa é sua, e totalmente privada.

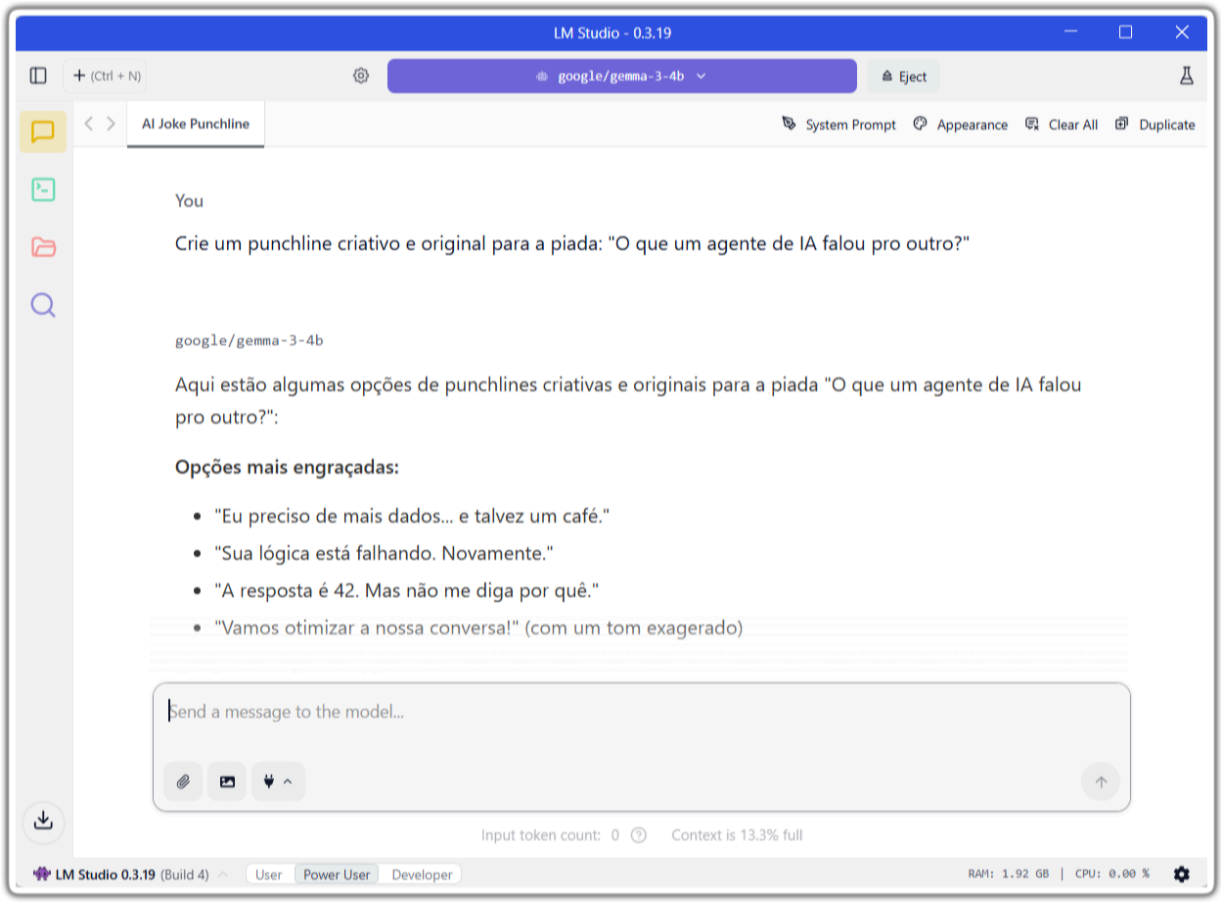

Figura 10.5: Exemplo de conversa com o Gemma 3 no LM Studio

10.4 Usando o modelo como servidor local

Se você for uma pessoa desenvolvedora e quiser usar essa IA local nos seus próprios projetos (integrando com Python, Node.js ou frameworks como *LangGraph*, por exemplo), o *LM Studio* tem um truque genial para isso.

Vá para a aba do "*Developer*". Com um simples clique no botão de ativar ao lado de "*Status: Stopped*" no canto superior esquerdo, o *LM Studio* transforma seu computador em um minisservidor de IA. Ele cria uma *API* local que imita perfeitamente a *API* da *OpenAI* ou da *Google*. Isso significa que você pode pegar um código em Python

que usaria o algum modelo *GPT* ou *Gemini*, apontar para o seu endereço local (indicado depois do texto "*Reachable at*"), e ele vai funcionar com o seu *Gemma 3*, de graça e offline. É simplesmente fantástico.

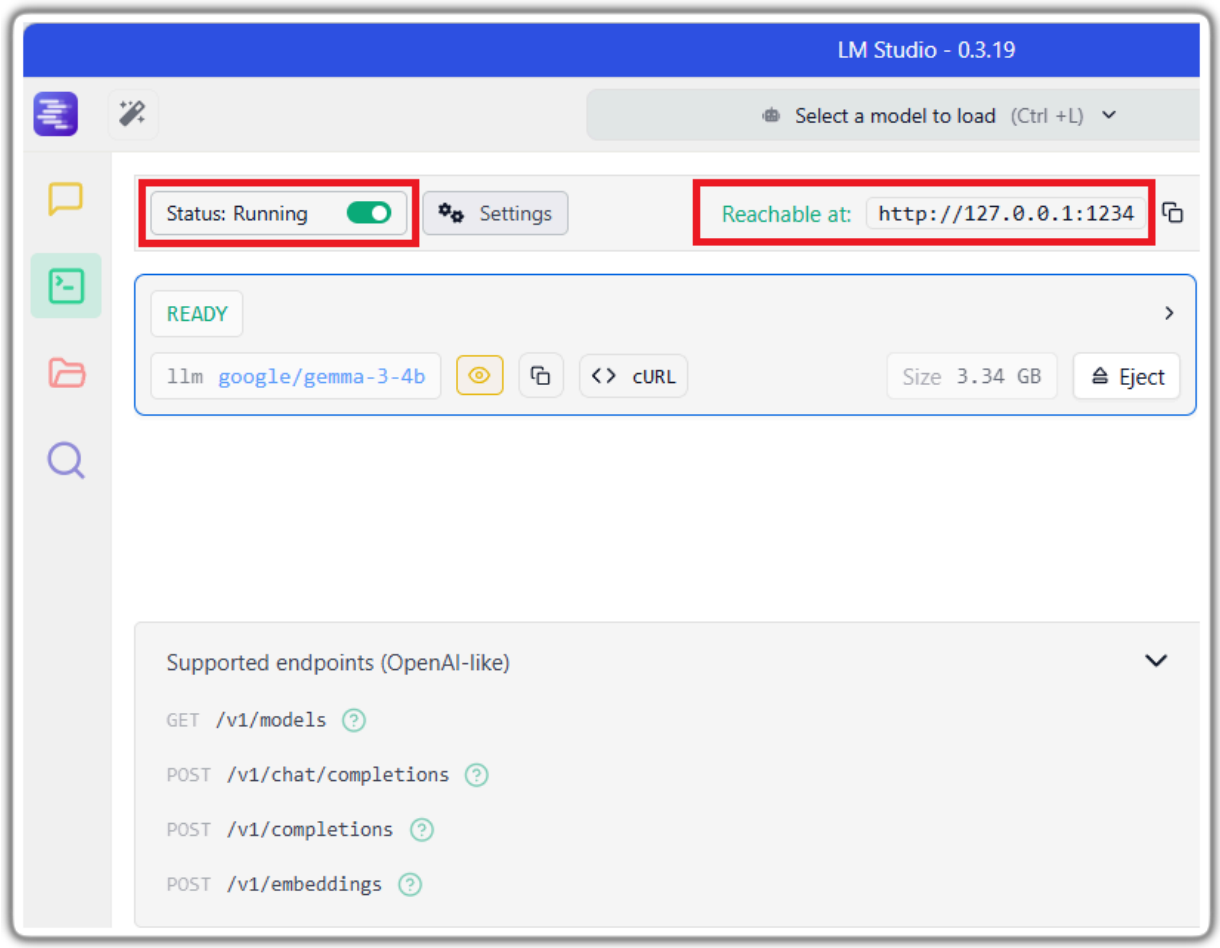

Figura 10.6: Usando o modelo Gemma 3 como API local LM Studio

Pronto! Com alguns passos simples, você saiu de um usuário de IA para um verdadeiro "hospedeiro" de IA. Você tem um modelo poderoso rodando na sua máquina, com total controle sobre seus dados e sem gastar um centavo.

Lembre-se de manter o *LM Studio* atualizado, pois a comunidade é superativa e novas melhorias e modelos chegam toda semana. E quem sabe, comece a participar da comunidade *open source*

também. Afinal, ter esses modelos rodando no seu próprio computador é como ter o seu próprio laboratório de IA na sua mesa. É o ambiente perfeito e seguro para você experimentar sem medo: testar um *fine-tuning* com *QLoRA*, brincar com *RAG* usando seus próprios documentos, ou simplesmente ver até onde a criatividade de um modelo pode ir, tudo isso sem se preocupar com custos de *API* ou com sua privacidade.

O que antes era um privilégio de gigantes da tecnologia, agora está acessível a qualquer entusiasta com um bom computador. É a democratização da IA acontecendo em tempo real, impulsionada por pessoas como você. Quem sabe o próximo modelo incrível, ou a próxima técnica revolucionária, não começa como um experimento no seu próprio PC? A porta está aberta.

Capítulo 11
Extra — Programando sistemas em Python com a API do GPT

Este é um capítulo extra apenas para pessoas desenvolvedoras ou interessadas no tema. Vou ensinar passo a passo como desenvolver um sistema de *chatbot* inteligente utilizando a linguagem de programação Python conectada à API da *OpenAI*.

Para isso, vou utilizar o *Google Colab* (https://colab.research.google.com/), um serviço de *Jupyter Notebook* do *Google* que permite programar e executar seus códigos em Python na nuvem, mas você pode usar a sua IDE preferida se quiser.

Ao abri-lo pela primeira vez, você verá uma tela parecida com esta:

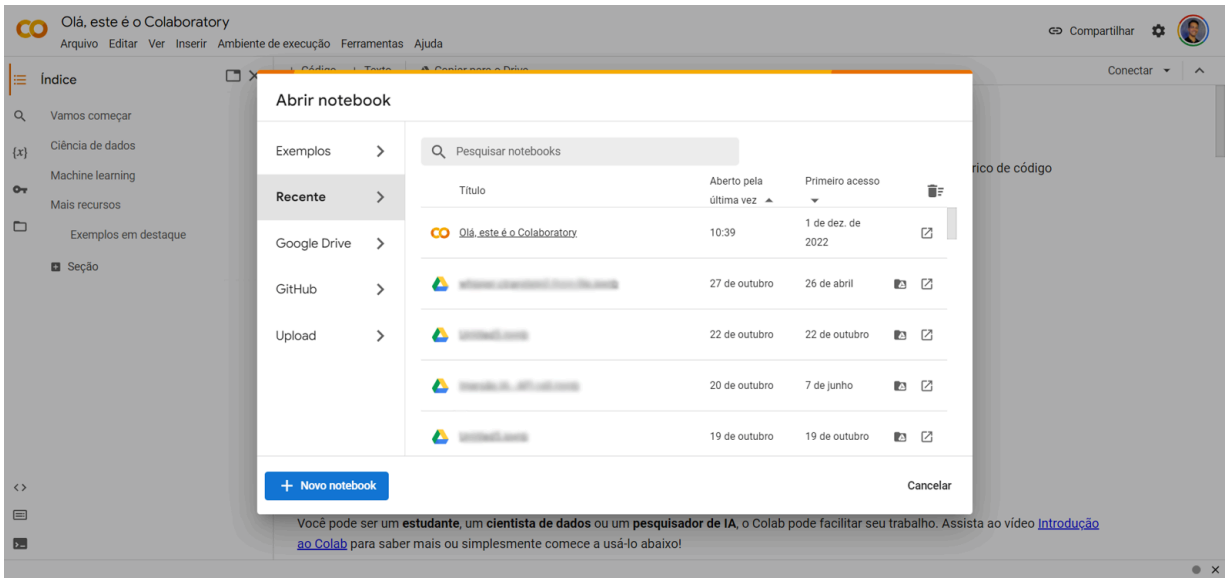

Figura 11.1: Tela de início do Google Colab

Clicando no botão *Novo notebook*, um novo projeto será criado. (Se você não vir essa tela, basta ir à aba *Arquivo* e selecionar a opção *Novo notebook*.)

Agora, podemos começar. Após clicar no botão de *+ Código* (logo abaixo da aba *Arquivo*), um novo campo de código será criado, e aqui, faremos a instalação da biblioteca da *OpenAI* no nosso *notebook*. Para isso, basta usar o código:

```
!pip install openai
```

Ao digitar isso, você notará um botão de *Executar* do lado esquerdo dessa célula:

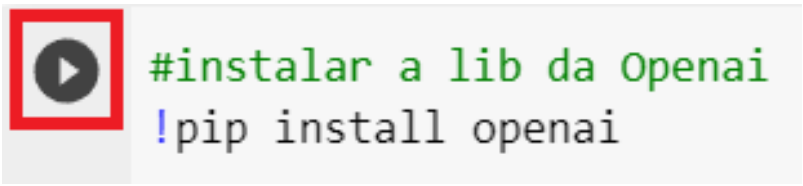

Figura 11.2: Botão de "Executar" em um campo de código

Ao clicar nele, essa célula será executada, ou seja, a biblioteca da *OpenAI* será instalada nessa instância do seu *notebook*. Após essa instalação, podemos criar um novo bloco de código e vamos importar a biblioteca:

```
import openai
```

Na sequência, vamos definir a *API key*, que é a sua conexão com a plataforma da *OpenAI*. Para isso, você terá que acessar a plataforma (https://platform.openai.com/), clicar no seu perfil e na opção *View API keys* (você terá que cadastrar o seu cartão de crédito para ter acesso a essa aba, pois as requisições à API da *OpenAI* são pagos).

Na nova tela, você poderá criar uma nova *key*, mas lembre-se de copiá-la antes de fechar a página, do contrário, terá que criar uma nova (e lembre-se também de NUNCA compartilhar sua chave de API com ninguém).

Para defini-la, você fará:

```
openai.api_key = 'INSERIR AQUI SUA API KEY ENTRE ASPAS'
```

Agora, vamos criar a variável que terá todas as mensagens, seja o comportamento que esperamos do modelo, nossos *prompts* e as futuras respostas. A API da *OpenAI* não tem (ainda, da data de escrita deste livro) um sistema de manutenção de estado, então temos que criar uma lista para manter todas as mensagens.

Essa lista será criada com um elemento, um dicionário com dois pares chave-valor: o `role` se refere ao dono da mensagem (aqui, será `system`, pois estamos definindo o comportamento geral do sistema), e o `content`, que é o conteúdo da mensagem (aqui, é o comportamento que o modelo assumirá):

```
messages = [
        {"role": "system", "content": "Você é um assistente prestativo."}
]
```

Seguindo, usaremos um `input` para esperar pela primeira pergunta ou *prompt* do usuário. Após recebê-la e armazená-la em uma variável, usaremos o método `append()` para adicioná-la à nossa lista de mensagens. Porém, note que agora o `role` é `user`, já que se trata de uma mensagem do usuário:

```
input_message = input('Esperando input: ')

messages.append({"role": "user", "content": input_message})
```

Terminada essa primeira parte, precisaremos de um *loop*, já que estamos criando um *chatbot* inteligente com memória das conversas. Para simplificar, vamos executar o programa sem parar até o usuário digitar a palavra "fim":

```
while input_message != 'fim':
```

E dentro desse *loop* `while`, teremos a nossa lógica de requisições à API. No pedido, passaremos o modelo que queremos usar (no caso, o *GPT-4*, mas poderia ser outro), o nosso histórico de mensagens, o número máximo de *tokens* e a temperatura (lembra desses

parâmetros que vimos no capítulo 6?), e guardaremos o resultado dessa chamada em uma variável `response`:

```
response = openai.chat.completions.create(
    model = 'gpt-4',
    messages = messages,
    max_tokens = 200,
    temperature = 1
)
```

Como podemos ver na documentação da *OpenAI* (https://platform.openai.com/docs/api-reference/chat), o objeto de resposta vem com uma lista de possíveis escolhas (geralmente, apenas a primeira posição, `[0]`, está preenchida). Acessamos essa posição da lista e, dentro dela, os campos `message` e `content` na sequência, que será o conteúdo da resposta em si.

A seguir, faremos novamente o `append()` dessa resposta no nosso histórico de mensagens, sendo que agora, como isso estamos adicionando a resposta do assistente, o `role` será `assistant`:

```
answer = response.choices[0].message.content

messages.append({"role": "assistant", "content": answer})
```

Por fim, podemos imprimir essa resposta obtida do *GPT-4*, esperar por uma nova mensagem do usuário (que pode ou não ser a palavra "fim") e adicioná-la ao histórico de mensagens com o `role` de `user`:

```
print("Resposta: ", answer, '\n\n')

input_message = input('Esperando input: ')

messages.append({"role": "user", "content": input_message})
```

Tchã-rã! É só isso! Você acabou de criar o seu primeiro *chatbot* inteligente usando a API do *GPT-4* da *OpenAI*, parabéns!

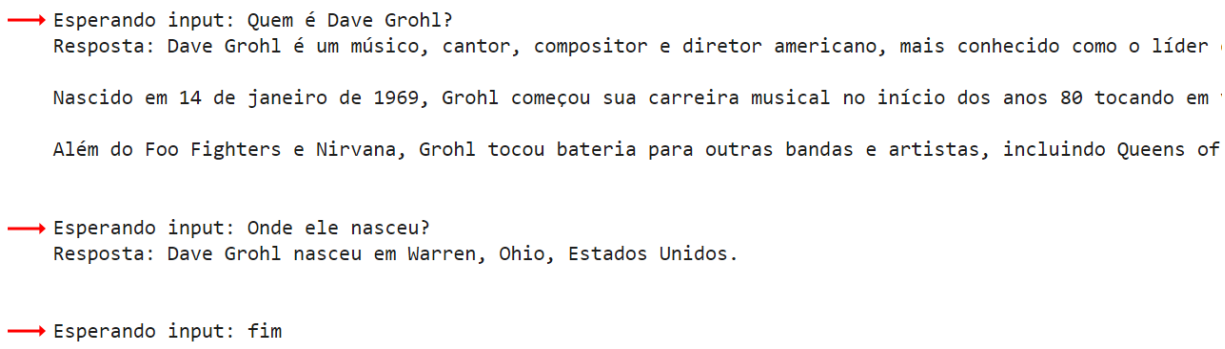

Figura 11.3: Exemplo de conversa com o chatbot criado

Note que, no segundo *input*, eu questiono "Onde **ele** nasceu?" e, como estamos mandando sempre o histórico inteiro de mensagens para a API, ele sabe, baseado no *prompt* e na resposta anteriores, quem é o "ele" da pergunta.

Apenas um cuidado a se tomar é que, como estamos sempre mandando o histórico inteiro e o pagamento é feito com base na quantidade de *tokens*, a conta pode acabar ficando cara para conversas muito longas. Para contornar isso, poderíamos colocar alguma restrição na quantidade de mensagens que podem ser adicionadas ao histórico, removendo as mais antigas quanto esse limite for ultrapassado. Isso fica como desafio para você (e você pode tentar resolvê-lo com a ajudar do *ChatGPT*!).

Com essa base, você já está pronto(a) para criar outros sisteminhas inteligentes usando a API da *OpenAI*.

Caso deseje, o código inteiro que criamos está no meu repositório do *GitHub* (https://github.com/fabriciocarraro/ia-python-openai) ou a seguir:

```
#instalar a lib da Openai
!pip install openai

import openai

# seta a API key
```

```
openai.api_key = 'INSERIR AQUI SUA API KEY ENTRE ASPAS'

# Define persona do sistema e cria histórico de mensagens
messages = [
        {"role": "system", "content": "Você é um assistente prestativo."}
]

input_message = input('Esperando input: ')

messages.append({"role": "user", "content": input_message})  # adiciona a
primeira pergunta no histórico do chat

# enquanto o input não for a palavra 'fim'
while input_message != 'fim':
    # Chamada à API
    response = openai.chat.completions.create(
        model = 'gpt-4',
        messages = messages,
        max_tokens = 200,
        temperature = 1
    )

    # Extrai a resposta de dentro do JSON de resposta
    answer = response.choices[0].message.content

    messages.append({"role": "assistant", "content": answer})  # adiciona
a resposta no histórico do chat

    # Imprime a resposta
    print("Resposta: ", answer, '\n\n')

    input_message = input('Esperando input: ')

    messages.append({"role": "user", "content": input_message})  #
adiciona a nova pergunta no histórico do chat
```

Espero que você tenha aprendido muito e se divertido lendo essas páginas, e que continue explorando o maravilhoso mundo da

Inteligência Artificial!